

Ronal Bejarano

**DESIGN AND IMPLEMENTATION OF A
HUMAN-ROBOT COLLABORATIVE
ASSEMBLY WORKSTATION IN A
MODULAR ROBOTIZED PRODUCTION
LINE**

Faculty of Engineering and Natural Sciences
Master of Science Thesis
October 2019

ABSTRACT

Ronal Bejarano: Design and implementation of a human-robot collaborative assembly workstation in a modular robotized production line

Master of Science Thesis

Tampere University

Automation Engineering. Factory automation and industrial informatics

October 2019

Over the last decades, the Industrial Automation domain at factory shop floors experienced an exponential growth in the use of robots. The objective of such change aims to increase the efficiency at reasonable cost. However, not all the tasks formerly performed by humans in factories, are fully substituted by robots nowadays, specially the ones requiring high-level of dexterity. In fact, Europe is moving towards implementing efficient work spaces where humans can work safely, aided by robots. In this context, industrial and research sectors have ambitious plans to achieve solutions that involve coexistence and simultaneity at work between humans and collaborative robots, a.k.a. “cobots” or co-robots, for permitting a safe interaction for the same or interrelated manufacturing processes. Many cobot producers started to present their products, but those arrived before the industry have clear and several needs of this particular technology. This work presents an approach about how to demonstrate human-robot collaborative manufacturing? How to implement a dual-arm human-robot collaborative workstation? How to integrate a human-robot collaborative workstation into a modular interconnected production line? and What are the advantages and challenges of current HRC technologies at the shop floor? by documenting the formulation of a human-robot collaborative assembly process, implemented by designing and building an assembly workstation that exemplifies a scenario of interaction between a dual arm cobot and a human operator, in order to assemble a product box, as a part of a large-scale modular robotized production line. The model produced by this work is part of the research facilities at the Future Automation Systems and Technologies Laboratory in Tampere University.

Keywords: Human-Robot Interaction, Human-Robot Collaboration, cobots, Industrial applications

The originality of this thesis has been checked using the Turnitin OriginalityCheck service.

TIIVISTELMÄ

Ronal Bejarano: Design and implementation of a human-robot collaborative assembly workstation in a modular robotized production line

Diplomityö

Tampereen yliopisto

Automation Engineering. Factory automation and industrial informatics

Lokakuu 2019

Viimeisimpien vuosikymmenien aikana robottien käyttö teollisuusautomaation alan tuotantolaitoksissa on kasvanut eksponentiaalisesti. Muutoksen tavoitteena on ollut tehokkuuden lisääminen kohtuullisin kustannuksin. Kuitenkaan kaikkia aiemmin ihmisten suorittamia työtehtäviä ei ole täysin korvattu roboteilla, erityisesti korkeatasoista näppäryyttä vaativissa tehtävissä. Itse asiassa Euroopassa ollaan ottamassa käyttöön tehokkaita työtiloja, joissa ihmiset voivat työskennellä turvallisesti ja robottien avustamana. Tässä yhteydessä teollisuuden ja tutkimuksen aloilla on kunnianhimoiset suunnitelmat saavuttaa ratkaisuja, jotka sisältävät ihmisten ja yhteistyörobottien – niin sanottujen cobottien tai co-robottien – rinnakkaiselon ja samanaikaisuuden työssä, ja mahdollistaen turvallisen kanssakäymisen samoissa tai toisiinsa liittyvissä valmistusmenetelmissä. Monet cobottien valmistajat alkoivat esitellä tuotteitaan, mutta tämä tapahtui ennen kuin koko toimialalle oli selkiytynyt tarpeita tälle erityiselle teknologialle.

Tämä diplomityö esittelee toimintamallin ihmisten ja robottien yhteistyöhön kokoonpanoprosessissa. Tämä on toteutettu suunnittelemalla ja rakentamalla erilaisiin teollisuuslaitoksiin kopioitavaksi sopiva työpiste, joka havainnollistaa skenaarion ABB YuMi dual arm cobotin ja ihmiso-peroijan vuorovaikutuksesta tuotelaatikon kokoamisessa, osana laajempaa modulaarista ja robotisoitua tuotantolinjaa. Diplomityötä varten valmistettu mallityöpiste on käytettävissä tutkimustointaan Tampereen Yliopistolla, Future Automation Systems and Technologies Laboratoryssa. Lopuksi kokeellisten testien yhteenveto esittelee nykyaikaisia hyötyjä ja haasteita cobottien toteuttamisesta tuotantolaitoksissa.

Avainsanat: Ihmisen ja robotin vuorovaikutus, Ihmisen ja robotin yhteistyö, yhteistyörobotti, cobotti, teolliset sovellukset

Tämän julkaisun alkuperäisyys on tarkastettu Turnitin OriginalityCheck -ohjelmalla.

PREFACE

The future of industrial systems is an attractive topic for many people on different disciplines, but only innovating and applying scientific knowledge in context might be the key for successful research and development strategies with real impact on society.

This work is product of more than 12 months of collective effort supported by the faculty of Engineering and Natural Sciences at Tampere University, through the Future Automation Systems and Technologies Laboratory. I would like to thank the Republic of Finland for giving the opportunity to international students to come and develop our academic plans under exceptional conditions of equality, equity and support, transforming universities into strategic nodes able to foster effectively the development of human potential, regardless nationality, language or culture. University life in Tampere is an outstanding experience.

I would like to dedicate this work to Amelda Rodriguez, Jorge Bejarano and Ernesto Cardozo, who always trust on my capabilities and provided all the conditions necessary to develop myself, far beyond my real opportunities in Colombia. Additionally, this work includes the support of Professor Jose L. Martinez L., Anne Korhonen, Matti Aarnio, Borja Ramis, Luis Gonzalez and Wael Mohammed as part of FAST-Lab staff.

All my gratitude to Dr. Jose Guarnizo, Dr. Nelson Diaz and Ana Triana who selflessly supported me on this endeavor. To my classmates, partners and coworkers Antonios Sylari, Saigopal Vasudevan and Dharmendra Sharma. Also, a special acknowledgement to Veera Niemi and all the Spinni crew, who provided me a deep insight of Finnish culture.

"Spain does not need savants"

Pablo Morillo

Tampere, 10th October 2019

Ronal Bejarano

CONTENTS

| | | |
|-----|---|----|
| 1 | Introduction | 1 |
| 1.1 | Motivation | 1 |
| 1.2 | Problem statement and research questions | 2 |
| 1.3 | Objectives | 3 |
| 1.4 | Scope | 3 |
| 1.5 | Outline | 4 |
| 2 | Literature review | 5 |
| 2.1 | Human-Robot Interaction | 5 |
| 2.2 | Human role on future manufacturing and partially automated processes | 7 |
| 2.3 | Cobots and best industrial practices | 8 |
| 2.4 | Related works | 10 |
| 2.5 | Review summary | 11 |
| 3 | Proposal | 13 |
| 3.1 | Formulation of a demonstrative human-robot collaborative process | 13 |
| 3.2 | Deployment of a collaborative process | 17 |
| 3.3 | Integration methods for a human-robot collaborative workstation to FASTory | 25 |
| 4 | Implementation and results | 28 |
| 4.1 | The new FAST-Lab | 28 |
| 4.2 | Implementation of a human-robot collaborative workstation, suitable for the modular-case assembly process | 30 |
| 4.3 | Cooperative product classification | 35 |
| 4.4 | Backend integration | 38 |
| 4.5 | Human Machine Interface | 42 |
| 4.6 | Adjacent devices | 42 |
| 4.7 | Testing method | 45 |
| 4.8 | Results | 47 |
| 5 | Conclusions | 50 |
| 5.1 | Supported and future works | 51 |
| | References | 53 |
| | Appendix A Scripts | 59 |

LIST OF FIGURES

| | | |
|------|--|----|
| 2.1 | Illustration of collaborative versus cooperative task classifications | 6 |
| 2.2 | Interaction paradigms for HRI | 7 |
| 2.3 | ColRobot prototype - Aerospace use case | 11 |
| 3.1 | Exploded view of the modular case for collaborative assembling | 14 |
| 3.2 | Case plans for laser cut production | 15 |
| 3.3 | Use case diagram for the Human-robot collaborative assembly workstation | 17 |
| 3.4 | ABB YuMi IRB14000 | 18 |
| 3.5 | Virtual manipulator configurations self-generated by YuMi to reach a work piece | 20 |
| 3.6 | Error on automatic path generation by YuMi | 21 |
| 3.7 | Correct path generation by YuMi aided by previous pose sculpt | 22 |
| 3.8 | Enhancement of Cobot Kinematic Behavior (ECKB). Flow chart | 23 |
| 3.9 | Teaching method for face matching (ECKB and RMAT). Flow chart | 24 |
| 3.10 | Sketch of integration for agents in the modular production line | 27 |
| 4.1 | Initial structure of FASTory | 29 |
| 4.2 | Conceptual 3D model of the transformation projected in FAST-Lab | 31 |
| 4.3 | Slotted surface distribution design for the collaborative work space | 32 |
| 4.4 | Slotted surface design of the pallet for the modular case faces | 33 |
| 4.5 | 3D model of the customized design for YuMi smart gripper fingers | 33 |
| 4.6 | Side view of pedestal for YuMi | 34 |
| 4.7 | Plant cut views of pedestal for YuMi | 35 |
| 4.8 | User interface of vision system component in RobotStudio | 36 |
| 4.9 | Demonstrative graphic models for cooperative classification | 36 |
| 4.10 | Hidden line view of air nozzle for air blast paper separation method | 37 |
| 4.11 | Capture of the visual design for the Human Machine Interface in the col- laborative workstation | 43 |
| 4.12 | Flowchart for the testing process | 46 |
| 4.13 | Final implementation of the human-robot collaborative workstation at FAST- Lab | 47 |
| 4.14 | Comparative bar graph of results in accuracy, precision and process time . | 48 |
| 4.15 | Advantages and challenges of implementing cobots by exemplifying a real industrial scenario of human-robot collaboration for assembly | 48 |

LIST OF TABLES

| | |
|---|----|
| 2.1 Overview of the relevant literature for human-robot collaborative manufacturing | 12 |
| 3.1 Assembly procedure of a modular case | 16 |
| 3.2 Technical characteristics of ABB YuMi IRB14000 | 18 |
| 3.3 General assessment of communication technologies agents | 26 |
| 4.1 Bill of materials for collaborative work space design. | 34 |
| 4.2 Vision system at YuMi. PatMax pattern settings | 37 |
| 4.3 Action codes for YuMi | 39 |
| 4.4 Description of events for Web APIs linked to the collaborative workstation . | 39 |
| 4.5 Web API for services provided by the collaborative workstation | 40 |
| 4.6 Web API to subscribe for events reported by the collaborative workstation . | 40 |
| 4.7 List of dependencies required on the virtual environment for the backend . | 41 |
| 4.8 Interactive events for the HMI | 44 |
| 4.9 Description of services and events for Web APIs linked to the side conveyors | 45 |
| 4.10 Web API for services and events provided by the conveyors | 45 |

LIST OF ANNEXES

| | | |
|-----|--|----|
| A.1 | Script for Right manipulator in YuMi (RAPID) | 59 |
| A.2 | Script for Left manipulator in YuMi (RAPID) | 66 |
| A.3 | Application backend (Python 3) | 69 |
| A.4 | Human Machine Interface - Frontend (HTML) | 81 |
| A.5 | Plans conveyor logic (ST) | 86 |
| A.6 | Pallet conveyor logic (ST) | 86 |

LIST OF SYMBOLS AND ABBREVIATIONS

| | |
|---------------------------------|---|
| ABB | Asea Brown Boveri (Swiss-Swedish multinational corporation) |
| AMIR | Autonomous Mobile Industrial Robot |
| ANSI | American National Standard Institute |
| API | Application Programming Interface |
| cobot | Collaborative robot |
| CSS | Cascading Style Sheets |
| e.g. | Exempli gratia (Latin term to abbreviate "For example") |
| ECKB | Enhancement of Cobot Kinematic Behavior |
| FAST | Future Automation Systems and Technologies |
| FASTory | Modular robotized production line at the Future Automation System and Technologies Laboratory of Tampere University |
| FoF | Factories of the Future |
| H2020 | Horizon 2020 (Framework programme funding research, technological development, and innovation by the European Commission) |
| HMI | Human-Machine Interface |
| HRC | Human-Robot Collaboration |
| HRCM | Human-Robot Collaborative Manufacturing |
| HRI | Human-Robot Interaction |
| HTML | Hypertext Markup Language |
| IEEE | Institute of Electrical and Electronics Engineers |
| IO | Input - Output (Interface for discrete control signals) |
| ISO | International Organization of Standardization |
| L ^A T _E X | a document preparation system for scientific writing |
| OKD-MES | Open Knowledge-Driven Manufacturing Executing System |
| PAP | Partially Automates Processes |
| RMAT | Reverse-Matching Assembly Technique |
| RTU | Remote Terminal Unit |
| SI system | international system of units (Système international d'unités in French) |
| TAU | Tampere University |

| | |
|--------|---|
| TCP | Tool Central Point |
| TCP/IP | Transmission Control Protocol / Internet Protocol |
| TUNI | Tampere Universities |
| UML | Unified Model Language |
| URL | Uniform Resource Locator |

1 INTRODUCTION

The role of humans on manufacturing industry changed repeatedly along history. Industrial manufacturing evolves rapidly, boosted by the deeper penetration of automation and information technologies through the entire life cycle of factories [21]. Time by time, human presence on the plant floor depletes, while robot manipulators and interconnected automation devices are more common, featuring an important relief for humans on heavy and repetitive mechanical tasks. Nonetheless, humans are still essential in many industries, due their high flexibility, intellectual and fast reasoning capabilities. Then, instead of pursue fully automated processes on industry, where costs and complexity factors might turn too high for certain productive sectors, new scenarios come out, where is optimal to have a safe mix between humans and robots, supporting the ideal of human-robot collaboration [57].

The concept of Human-Robot Collaboration (HRC) represents a complementary skill sharing, providing the cognitive and flexible kinematic skills from humans, together with capabilities of intensive, accurate and reliable repetitive manipulation, provided by robots. In practice, the industrial standards related to collaborative robotics, give guidelines to ensure safe coexistence for humans during the execution of simultaneous tasks with robots [20].

1.1 Motivation

Nowadays, HRC is a relevant research topic, with more than 1.500 references on scientific document resources just from IEEE. Interest about this topic comes also from industry, with an increasing number of factories planning or already migrating their human-centred workstations, to robotized workstations, without investing additional resources for safety conditioning (protective barriers, indicators etc). In addition, the participation of Tampere University on research and innovation projects for Factories of the Future (FoF), such as Zero Defects Manufacturing Platform, requires research and test environments, able to demonstrate Industry 4.0 concepts in multiple scenarios, including collaborative robots, also known as *cobots*. All together, creates high expectations around the HRC issue from important players on research and development, with clear impact on industry.

Integrating the concept of HRC and manufacturing has many benefits. Mainly, it can reduce the complexity of manipulators for high dexterity tasks, as well as simplify reasoning

capabilities of automated control systems. The adverse effects of using complex electromechanical systems on robot manipulators, might have a negative impact in plant floor performance indicators, such as maintenance periods, failure rates, manufacturing time and energy efficiency [29]. On the other hand, by using HRC, the models of manufacturing processes might be susceptible of high uncertainty, reducing their prediction accuracy due the aleatory nature of human behavior. Then, the robots might require to be aware of their environment, to adapt properly and interact autonomously and effectively with humans. This last aspect, might be a drawback of HRC, which can add complexity to any Human-Robot Collaborative Manufacturing (HRCM) system, demonstrating that still there is room for improvement on this matter.

The penetration of cobots and HRCM in industry is still premature, although, the cobot industry revenue is expected to increase more than 24 times by 2025, compared with numbers from 2017 [14]. This situation provided a research motivation to the academic team at the Factory Automation Systems and Technologies Laboratory (FAST-Lab) in Tampere University, where topics as Factories of the Future (FoF), Industry 4.0 and the Internet of Things are boarded from several research projects.

Ongoing projects at FAST-Lab are based on pillars for digitizing and transform the European industries, funded by the European Council on the framework of the program Horizon 2020 (H2020). Those projects require especial experimentation facilities, where multiple technologies converge to provide demonstrative scenarios for research and development. The enhancement of the laboratory for that purpose, include the creation of new research spaces, interconnected by using Information and Communication Technologies (ICT), but at the time, centered on the human potential on future automation. For that reason, the creation of a human-robot collaborative workstation was conceived, intended for host multiple research projects around FoF topics, including HRC and HRCM.

1.2 Problem statement and research questions

Preparing FAST-Lab for new research and development projects by creating a human-robot collaborative workstation requires to answer the following specific research questions:

- How to demonstrate HRCM?
- How to implement a human-robot collaborative workstation?
- How to integrate a human-robot collaborative workstation into a modular interconnected production line?
- What are the advantages and challenges of current HRC technologies at the shop floor?

1.3 Objectives

The main objective of this work is to design and implement a human-robot collaborative assembly workstation, integrated to the FASTory modular robotized production line at FAST-Lab. Another specific objectives are:

- Select and design a demonstrative human-robot collaborative assembly process.
- Design and implement a human-robot collaborative assembly workstation able to reproduce the process selected.
- Design and implement methods to integrate the human-robot collaborative workstation to the FASTory modular robotized production line.
- Asses the advantages and challenges of current HRC technologies.

1.4 Scope

The present work extents to the following conditions established by the resources available in FAST-Lab:

- All designs are based on the dual arm cobot *YuMi* from ABB (IRB-14000), provided by Tampere University at FAST-Lab.
- The designs of the human-robot collaborative workstation include structural specifications referenced on international standards and commercial products.
- The collaborative process will be explained by illustrative images and modelled using Unified Modeling Language (UML). The work pieces used on it, might include assisted designs, provided by free software tools.
- The development of the control settings and programming for the cobot required for the collaborative process, is based on ABB Robot Studio 6.07 and RAPID programming language.
- The methods to integrate the human-robot collaborative workstation are based on RestFul web services and TCP/IP Web Sockets. Its implementation is based on Python 3, HTML and CSS programming languages.
- The human-robot collaborative workstation might have interconnections to the FASTory line and the industrial dual arm robotic cell (ABB IRB-140) through the use of a Mobile Industrial Robot model MiR-100 and conveyor belts, respectively. All the interconnected systems are compatible with the technologies aforementioned for the integration methods.

Any additional component not listed in this section will remain on consideration of the author to be included or proposed as future work.

1.5 Outline

This document will present initially on chapter 2, a literature review section, result of researching the state of the art of HRC, human-robot interaction and cobots. Chapter 3 presents a proposal section, documenting the methodology to carry the collaborative process, the workstation and the integration methods. Then, chapter 4 will present the implementation phase, an experimental method to asses HRCM based on the workstation implemented and the results of implementing a workstation. Finally, chapter 5 will present the conclusions of the work, summarizing the results and proposing future works.

Part of this work was compiled and published through the document *"Implementing a Human-Robot Collaborative Assembly Workstation"* presented on the IEEE International Conference on Industrial Informatics INDIN'19 in Espoo, Finland. It was awarded among the best presentations of the conference.

2 LITERATURE REVIEW

Oftenly, technical definitions of robots are based on early concepts of robotics, developed exclusively for industrial automation applications. In synthesis, a formerly concept of robots defined them as automatic, reprogrammable, multipurpose manipulators [9]. Nevertheless, humans started to be surrounded by an increasing presence of automated devices with cognitive features, usually integrated to robotic autonomous systems. Then, the interaction and coexistence of both entities in renovated environments started to be matter of study, specially because of the impact on redesigning work spaces traditionally exclusive for humans, to robotic cells supervised, commanded and maintained by humans. Aspects about technical and social dimensions for these new environments shaped the current concept of Human-Robot Interaction (HRI).

This section will present an insight of HRI, the role of humans on future manufacturing and the state of the art on industrial collaborative robots, based on the latest international standards. Additionally, some other related works will be analyzed and summarized.

2.1 Human-Robot Interaction

The study of HRI focus on the influence of articulate reciprocal events between robots and humans. Scientific authors compile and categorize its attributes through defining a taxonomy. For instance, A. Yanco, proposed 11 categories to evaluate the possible scenarios of humans and robots working together, including level of shared interaction among teams, interaction roles, type of human-robot physical proximity, time/space taxonomy and autonomy level/amount of intervention [65]. In contrast, G. Michalos, gives more relevance to assess the work space and task participation [35]. Together, the approaches raise the relevance of categories related to use of the work space and active time ratios.

Probably, the most prevalent classification parameters on HRI taxonomy, associate time and space with physical and cognitive interaction. For HRC assembly tasks, there is simultaneous coexistence of human-dependent processes. Other cases as human-robot cooperative tasks can classify as shifted coexistence of independent processes. In this context, cooperative (switched) tasks require at least one participant on idle state at the time, while the counterpart works, whereas collaborative tasks, require the robot to handle process parts statically or dynamically, while the human manipulates the work piece

simultaneously (Figure 2.1).

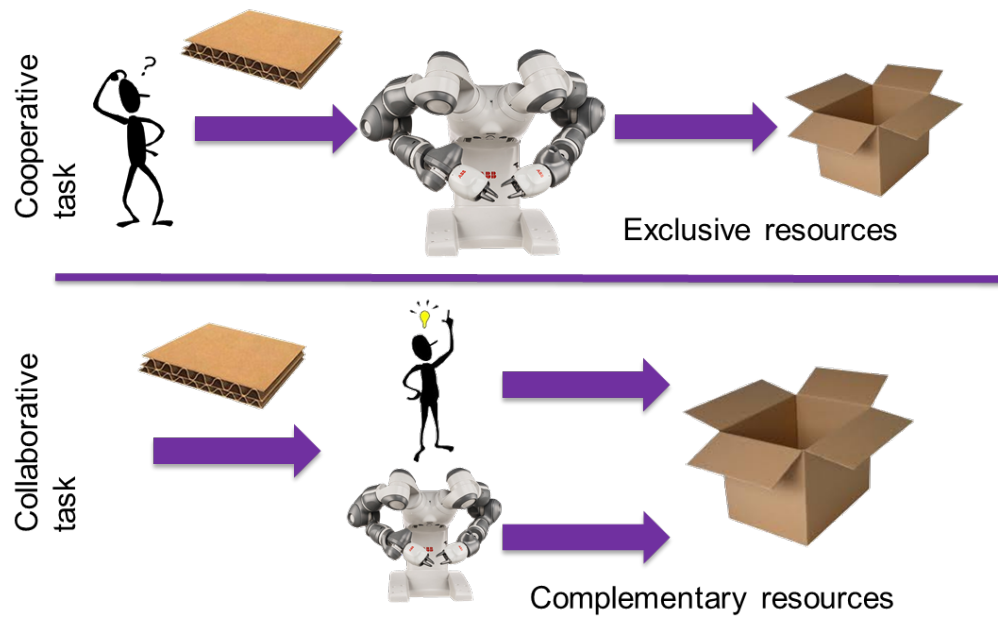


Figure 2.1. *Illustration of collaborative versus cooperative task classifications*

Interaction paradigms can also help to depict HRI. Interactions controlled by humans as the one presented by H. Tang et al. performing human-robot collaborative teleoperation for reconnaissance robot [54], contrast with autonomous applications as the one presented by L. Rozo et al. about collaborative robot behaviors from human demonstrations [43].

Figure 2.2 presents a simplified classification of paradigms on HRI, inspired on cognitive characteristics. The collaborative interaction defined in this study case, is allocated closer to autonomous paradigms, following a model of robot – human activity, commanded under safety constraints.

Lately, the research goals on robotics reshaped to adapt and match diverse paradigms and attributes of HRI, encouraged by the popularization of specialized robots on industrial or service/social purposes, operating on human environments. While industrial HRI focused on control, safety for human workers and process optimization, as seen on [9], [61] or [39], service/social HRI focused on behavioral awareness and comfortable interaction of humans aided by robots, as presented in [5] or [8].

Despite certain differences between industrial and social HRI, some considerations related to psychosocial human behavior might influence industrial HRI from a social approach. For example, S. Stadler et al. describes in [53] the influence of the appearance of robots over the expectations of robot users. Human-like (anthropomorphic) or machine-like (functional) manipulators might impact on the user behavior. This aesthetic bias can change the human perception about the robot. It was demonstrated that people associates anthropomorphic robots as a possible work partner, better than functional models. Another study presented by D. Bortot et al. about human behavior influencing

industrial HRI, confirmed the hypothesis that robots moving near humans in rectilinear displacement paths, create a human well-being perception, since humans feel in control when can determine predictable dynamic situations [7]. These studies give important clues to improve industrial HRI on collaborative work scenarios, based on analyzing human behavior and demonstrating both paradigms can complement each other enabling HRC.

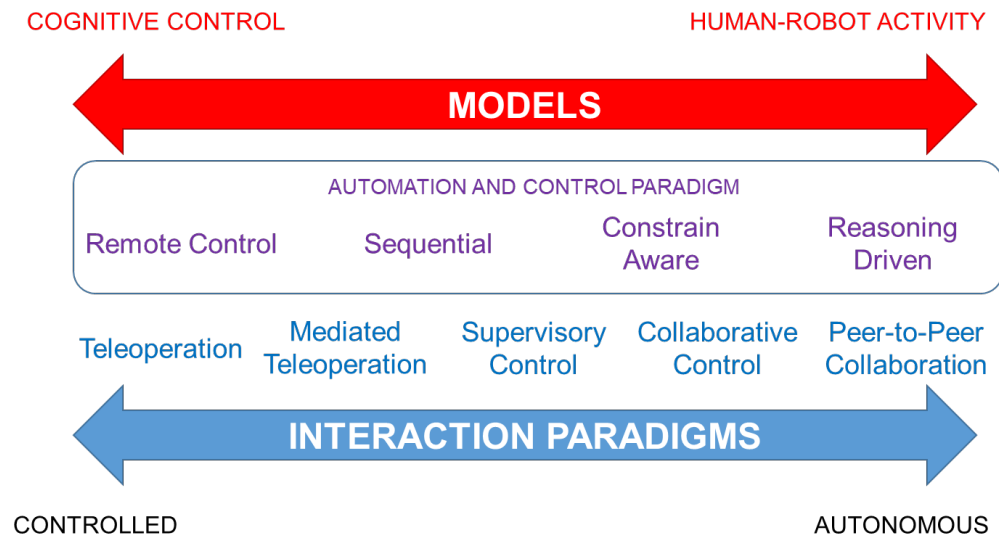


Figure 2.2. Interaction paradigms for HRI, based on Figure 71.11 from [47]

2.2 Human role on future manufacturing and partially automated processes

In a contemporary industrial environment, large-scale manufacturing tend to rely on automated processes by means of devices such as controllers, instrumentation, robots and Human-Machine Interfaces (HMI). Social impact of automated robots usually carries the concern about loss human jobs, since process optimization and technologies as Artificial Intelligence (AI), enhance cyber-physical systems to have certain level of reasoning and predictive abilities comparable, or sometimes even better than human competences. This situation leaves in some cases human skills relegated and obsolete on reasoning, heavy, repetitive or precision tasks [30]. However, this situation is not new on human history. Previous milestones on history marked a series of periods known as industrial revolution. Such periods included changes on manufacturing systems by mechanizing manual tasks, inducing a radical change on their technological structures, but giving new opportunities and better living standards to the working class people. Since then, population could focus on areas with superior impact and spend time on personal development.

Predictions about the role of humans in future manufacturing augur to be exclusively related to creative skills and other occupational areas with imperative need of social intelligence and empathy, e.g. education, hospitality, healthcare and tourism [46]. On the other

hand, occupational expectancy for technicians follows the growing demand of experts in robotics and AI.

By exploring novel advances of robotic processes towards the future of humans in manufacturing chains, new technologies considered few years ago as a concept of the future, are nowadays operating commercially under supervision of humans. Those technologies such as remote ship operation, monitoring of flexible manufacturing lines, distributed automated control and protection of electric power transmission lines, usually require limiting intentionally the mechanical scope of machinery, for safety mainly, but also for quality or just user comfort purposes. These processes always require room for activities exclusively performed by humans. Those cases could fit on the classification of Partially Automated Processes (PAP), which is the best category identified to target a survey of HRC.

Justify the dependency of humans in manufacturing can be complex, it is one of the major present challenges to disseminate cobots in factories, since conventional robots on the market can present powerful features compared to several mechanical skills of humans. Still, there are clear examples of PAP, where robotics found a place complementing human skills. Partially automated driving, is a step taken by car manufacturers to get closer around autonomous cars. Because of regulations and technological limitations, safe operation of autonomous cars is a feature not available in commercial models yet, but major improvements on driving experience are already integrated to new automobile models, as highway pilot, which merges cruise control and lane-keeping systems, creating a combined automatic function for assisted driving [12]. Conceptually, partially automated driving can represent a use case of HRI in a PAP. This example confirms the possibility of apply concepts of HRC on manufacturing despite the lack of motivation on industry about cobots perceived on this survey.

2.3 Cobots and best industrial practices

Cobots are manipulators intended for direct physical interaction with a human operator [39]. On industrial scope, the German Institute of Occupational Safety defines them as *“complex machines which work hand in hand with human beings”* . . . *“in a shared work process, they support and relieve the human operator”* [17]. Primitive models of cobots relied on humans to overcome quick reasoning and vision guided processes, as seen widely on car assembly processes such as the examples presented in 1997 by Wannasuphprasit et al. [61]. In that case, a manipulator carrying heavy parts for assembling was guided by a human, while a cobot simultaneously steered the manipulator guiding its trajectory through predefined surfaces to avoid collisions. Lately, technology covered reasoning and image processing by AI and Machine Vision (MV) applications as the one presented in 2005 by Lopez-Juarez et al. [31]. However, computing all the data required for the aforementioned technologies, demanded the use of exceptional computational resources, pushing to explore computing solutions based on networked computing clusters

(cloud), but creating a new vulnerability on security, already addressed by S. Olaiya et al. [2].

The technical features of cobots became matter of discussion for multiple scientific communities and escalated to be topic of international standardization, since the principles of coexistence and simultaneity of robots with humans prevail among cobot manufacturers and academics. Standards helped to establish descriptive features for collaborative robotics, to comply functional safety like any other end-user product [27]. The International Organization of Standardization (ISO) released the standard ISO 10218 on 2006 and updated on 2011, for safety requirements on robots (part 1) [9], together with robot systems/cells and applications (part 2) [28]. The American National Standard Institute (ANSI) adopted the standards releasing the standard ANSI/RIA R15.06-2012 on 2012 which specifies the requirements for collaborative operation [4]. The requirements on the standards included mandatory visual indication while the robot is in collaborative operation and one or more of the following features [28], [4]:

- *Safety-rated monitored stop (IEC 60204-1 Category 2)*: In presence of an operator or obstruction, the robot should ensure stop-motion, keep the electrical motor drives on and resume motion after the obstruction clears, without any additional action over the power supply or control cycle system. Additional methods to stop the mechanism should be available in case of stop condition violation.
- *Hand guiding (Emergency stop and enabling device)*: The robot should stop in presence of the operator and wait for enabling to activate the motion leaded by the operator through direct interface. This enables the operator to sculpt poses or teach spatial targets. Non-collaborative operation can continue after the operator leaves the work space.
- *Speed and separation monitoring*: Separation distances should be continuously monitored by scanners, vision systems or proximity sensors. Robot speed directly correlates to separation or distance between the operator and manipulator. Pre-defined zones in function of human and robot speed, reaction time of the robot/detection system, intrusion distance capability with extend extremities and position uncertainty can dictate the maximum allowable speed of the manipulator. Moreover, stop condition should activate, following the Safety-rated monitored stop requirements.
- *Power and force limiting*: Operation of the robot is limited in energy supply to the actuators, to avoid hazards for the operator. The manipulator design eliminates pinch points, sharp edges or any other physical characteristic risky for human direct contact. The robot should be able to detect and react at direct contact. This part requires a tailored risk assessment, since the hazard threshold may vary depending the part of the human body exposed to effective collision force applied by the manipulator. Different trajectories avoiding sensitive areas of human body and control methods for each cobot application can influence drastically the power and force limiting function.

Lately, ISO/TS 15066:2016 carried out the standard specifications for industrial collaborative robots, based on discussions from ISO 10218 - part1 [9], about variations on energy, speed thresholds and contact areas to avoid pain or injury on humans during incidental contact [24]. Nonetheless, concerns about the impact on production efficiency from complying safety arouse, especially because limiting the energy provided to the manipulator, also might constrain productive characteristics as payload or TCP velocity, reducing the flexibility of cobots by exempting them of executing any non-collaborative task easily performed by conventional robots.

2.4 Related works

Several experiments on automation and ergonomics focused on assess the capabilities and suitability of collaborative robotics on manufacturing scenarios. I. Makrini et al. presented on 2017 a technological structure for human-robot assembly tasks, including gesture recognition for control purposes, face recognition together with human-like robot behavior to integrate ergonomic metadata facilitating intuitive interaction and visual inspection for manufacturing process improvement. A Baxter dual arm cobot was used in this experiment to validate the structure [33].

Another experiment, carried by A. Cherubini et al. on 2015, assessed the use of cobots to assemble a car homokinetic joint (Rzeppa joint) able to transfer power between drive shafts trough a variable angle but constant speed, limiting mechanical friction and play. This task includes the insertion of spherical parts, which was in charge of operators, but it might generate Musculoskeletal Disorders (MSD) in long term, due the positions and loads manipulated on the process. Implementation included a work cell equipped with a collaborative robot (KUKA LWR IV) aided by machine vision application and following the standards available for speed monitoring besides power and force limitation. The experiment demonstrated that even the process cycle was longer, the burden for operators was lower, decreasing drastically the costs related to MSD, and opening the possibility of having rapid return of investment by improving the process speed in future works [13].

Collaboration between humans and robots analyzed by V. Villani et al. on 2018 presented safety, intuitive programming and interaction methods as the biggest limitations in the matter. In addition, the importance of increase the penetration of collaborative robots on manufacturing should guide future research on the field. While performance-oriented solutions can address current issues, such as safety, the costs of robots and capability to upgrade older technologies to collaborative features can be an opportunity to small and medium-size companies. Finally, a prospect to improve cognitive processing skills and shared autonomy capabilities can enable cobots to assist human workers not only by mechanical means but also cognitive effort reduction [59].

Another relevant research work is ColRobot, a concluded H2020 project coordinated by Ecole Nationale Supérieure D'Arts Et Métiers that proposed an integrated system for

collaborative robotics using a mobile autonomous manipulator to deliver tools, parts, prepare kits for assembly and hold work pieces for human operators. The project proposed HRI by cognitive and physical characteristics by using gestures, touch commands and demonstrations. A safety system that includes localization of the operators was integrated. The project was tested on two use cases, automobile and aerospace industry will be implemented and validated in real world operational environments. This project presented multiple developments on HRCM by publishing scientific documents about human behavior and hand gesture classification form smart HRI [34], 3D Metrology using a cobot with a laser triangulation sensor [51], minimum distance calculation for safe HRI [45] among others as [49], [10], [23], [38], [11], [22], [44]. Figure 2.3 presents the prototype of a single arm cobot installed over an autonomous mobile robot. This approach is widely used for research on autonomous collaborative assistantas for manufacturing.



Figure 2.3. *ColRobot prototype - Aerospace use case [15]*

The work presented by H. Wali based on Design For Assembly methodology (DFA) provided a similar framework than ColRobot prototype for HRCM, but demonstrated the assembly of a low voltage electrical installation mounted over a DIN Rail. This work provides an important reference to break up assembly task, model them mathematically and decide which are optimal to perform by the human, robot or in collaborative way. However this work uses a single arm robot (KUKA LBR IIWA 7 R800) mounted in a mobile robot (KUKA flexFELLOW H750 extended), for a nuclear workstation, not integrated to a production line [60].

2.5 Review summary

The documentary resources summarized in this section demonstrate the current interest from the scientific community on HRCM and HRI. Motivating aspects as the support given by the European Commission to continue innovation, research and development projects

about the main topics of this work, provide a promising scenario for a new human-robot collaborative assembly workstation in Tampere University. In parallel, it was possible to identify a clear trend of using multiple technologies from different manufacturers based on Standards as ISO 10218 2011, part 1 and 2, ANSI/RIA R15.06-2012. The rise of experimental prototypes including diverse commercial models leads to the conclusion that technology integration is a core technical topic for applied HRCM and HRI.

A compendium of related works is presented on Table 2.1, classified by the HRCM process implemented, all of them aiming to assembly. Most of the research works cited use a single arm cobot, this demonstrates that research with dual arm cobots is a novel topic. In addition the most common method for control is force detection, since some cobot models include smart detectors linked to automated tasks. However, in a real industrial environment is unacceptable to have accidental triggering of commands. This might require more research tasks in the future.

Table 2.1. *Overview of the relevant literature for human-robot collaborative manufacturing*

| Process | Type of robot | Control and integration interface |
|--------------------------------|-------------------------------------|---|
| Box assembly [33] | Rethink robotics - Baxter, dual arm | Gesture recognition by visual interface |
| Joint assembly [13] | KUKA LWR IV, single arm | Admittance controller - Force detection |
| Mobile assembly assistant [34] | KUKA LWR IV, single arm | Admittance controller - Force detection, visual recognition, multisensor handheld |
| Electric module assembly [60] | KUKA LWR IIV, single arm | Admittance controller - Force detection |

3 PROPOSAL

The Factory Automation Systems and Technology laboratory (FAST-Lab) is an experimental environment at Tampere University where automated devices are set up to operate in demonstrative industrial manufacturing processes. The laboratory has recently acquired new equipment to reinforce the teaching and research activities. These devices include an autonomous mobile industrial robot (MIR-100), an industrial dual-arm robot (Yaskawa SDA-10F), a collaborative robot (ABB YuMi cobot) and multiple stationary articulated robotic arms, all planned to merge into the existing robot cells in both the FASTory and FESTO production lines. The latter are conveyor-based production lines following different automation architectures and technologies.

This section will present the formulation of a demonstrative human-robot collaborative process. Then, the methodology to deploy the collaborative process in a cobot will be explained, based in offline and online robot programming. Finally, the methods to integrate the human-robot collaborative workstation to a modular production line will be addressed.

3.1 Formulation of a demonstrative human-robot collaborative process

Many processes are already performed by robots on industrial manufacturing, such as welding, packing, sorting, labeling, cutting etc. as explained on chapter 2. Nevertheless, safety conditions keep conventional robots away of humans. Finding a convergence point between humans and robots on specific manufacturing processes is not an easy task, since the robots are conceived as machines faster and stronger than humans. This perception can influence to reject any possibility of having a combination of both entities and be competitive with conventional robot models.

To solve the puzzle of finding a HRC demonstrative process, suitable for industry, a survey of strategic and unique skills between human and robots was made in FAST. Some early ideas pointed to build an assisted collaborative studio to aid the creation of visual art. That idea projected a cobot supported by a machine vision application, providing rectilinear isolation paths for artists using the geometric abstraction technique [16]. However, this idea was discarded by the project supervisor, because its technical complexity and low potential impact at industrial level. Also, local academic departments of plastic arts did not showed much interest to work on it.

Later, another approach, listed strengths of robots in manufacturing such as good capacity to reconfigure for dynamic processes, ability to perform intensive repetitive mechanical tasks and fast communication interfaces. Then, a parallel list focused on humans, highlighted the excel on adaptive cognition and high dexterity for grasping and manipulation. From the competences mentioned, it was concluded that assembling a special case structure might require to use strengths from both, humans and robots, in a simultaneous and coexistent way.

The selected structure for the modular case is a computer-assisted design of a six face polyhedron, provided by the free online tool *makercase* [32]. This design, exclusive for flat materials of 3mm thickness, includes tab - slot edges of 6 mm length and slots for fasteners near each corner, made of sets of a M3 nut and 10mm bolt. The internal dimensions of the case are 155mm by 110mm by 25mm suitable to carry a batch of more than 50 paper sheets caliper 0.002 of 105mm by 150mm (FASTory products). Figure 3.1 presents an exploded view of the case model and Figure 3.2 shows the case plans for laser cut production. This structure was selected due:

- *Minimal design*: The design is simple and easy to modify
- *Easy to produce*: Laser cut plans allow to produce prototypes faster than other methods as 3D printing
- *Robust structural features*: Exceptional integrity withstand under external forces
- *Complex assembly process*: The process to match faces and fastening small bolts might be over demanding for a single human

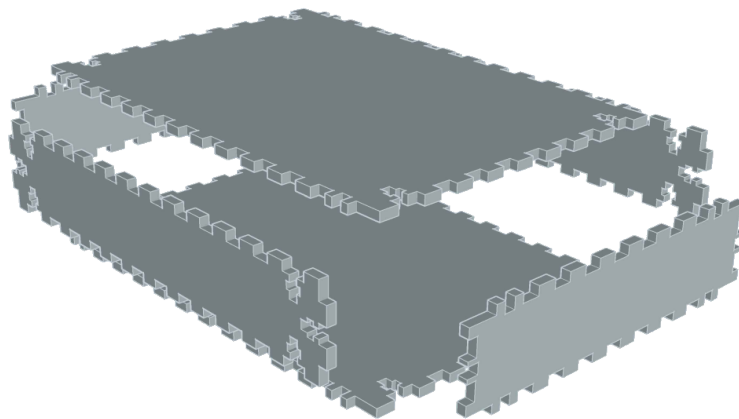


Figure 3.1. Exploded view of the modular case for collaborative assembling

By completing the design of the work pieces, it is possible to move forward and make a generic assembly procedure for the case. The procedure selected is inspired on the steps taken by a human to assemble the case, consisting on pick and attach one new face at the time. Since the design selected includes a set of 14, 9 and 4 tab-slot matching guides on each long, width and height edge, the accuracy and precision to assemble

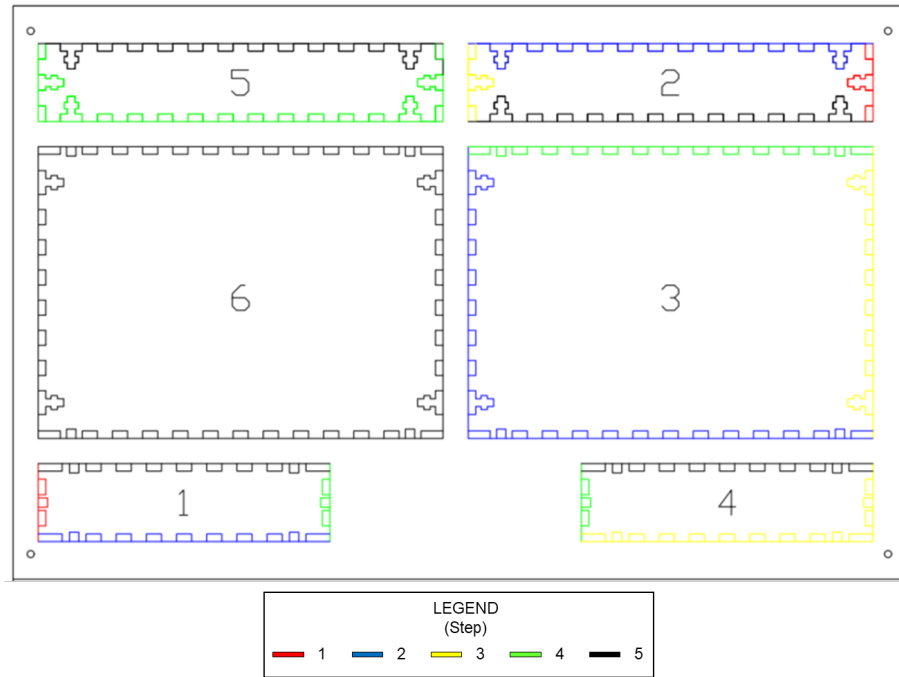


Figure 3.2. Case plans for laser cut production. Color marking on edges matched every assembly step

successfully the case is critical. As presented on Table 3.1, the procedure require to match correctly from 4 to 46 6mm tabs simultaneously.

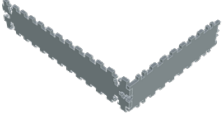
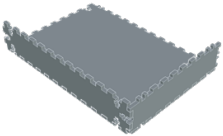
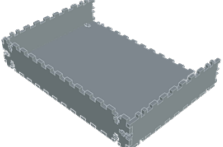
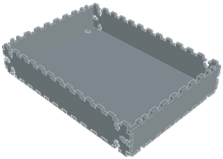
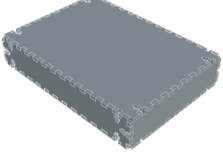
From this experimental observation, each face was labeled with sequential numbers (1 to 6), to guide the collaborative process. Figure 3.2 presents the case plans with enumerated faces and edges marked in colors, which corresponds to the section that should be matched on each assembly step.

Having an assembly procedure prepared, is possible to proceed with the general process definition. As soon as the AMIR arrives to the collaborative workstation, the product basket can be unloaded from it and placed on the collaborative work space. Then, the modular case parts can be loaded into a conveyor for further preparation. Both tasks are in charge of the robot. The conveyor leads the shipping box material to the dual robot cells, programmed to label and organize in one pallet all the faces of the modular case. Then, a conveyor belt transports the pallet to the collaborative station by a conveyor belt.

At this point is important to take three design parameters into account. First, the workstation should be able to perform collaborative and cooperative tasks sequentially. Second, the process flow on the collaborative tasks, should be controlled by the human. Third, the execution might require coordinated work between both arms of YuMi. This last to ensure accurate and precise matching between each face of the box. Those conditions should be integrated in the general process definition.

Having the parameters mentioned into account, the process can continue on the collaborative workstation, where a human operator loads the pallet into the work space and YuMi. Then, the procedure to assemble the modular case can begin. Afterwards, YuMi

Table 3.1. *Assembly procedure of a modular case for the collaborative process*

| Step | New faces attached | Number of fasteners | Remarks // tabs | Illustration |
|------|--------------------|---------------------|----------------------------------|---|
| 1 | 1, 2 | 1 | None // 4 |  |
| 2 | 3 | 4 | None // 23 |  |
| 3 | 4 | 3 | None // 14 |  |
| 4 | 5 | 4 | Ready for packing products // 17 |  |
| 5 | 6 | 8 | Sealing // 46 |  |

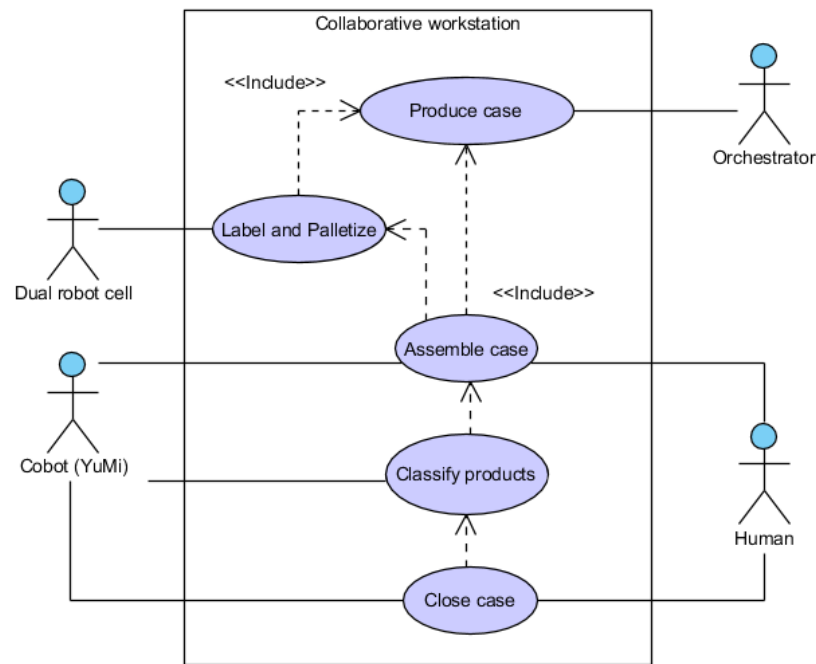


Figure 3.3. Use case diagram for the Human-robot collaborative assembly workstation

scan and classify the products using a built-in smart camera and suction cup in one of its hands, aided by an external blower to ensure single sheet grasping. To conclude, human and robot finalize the packing process by closing the box and dropping it to a delivery buffer. Here we can identify collaborative tasks for assembly and a cooperative task for the product classification made by YuMi. Figure 3.3 presents the use case diagram, including the activities held in the human-robot collaborative assembly workstation.

The human-robot collaborative assembly process presented, provides a demonstrative task, where human and robot perform actions simultaneously in the same work space. Additionally, a cooperative classification process was integrated, expanding the features of the workstation for demonstrative purposes.

3.2 Deployment of a collaborative process

The collaborative process proposed for assembly the special model of modular case requires high accuracy and precision, since 4 to 46 tabs-slot sets of 6 mm by 3 mm by 3 mm must be matched time by time. The cobot model for this specific use case is the collaborative dual-arm ABB YuMi (3.4). Technical specifications of YuMi are depicted on Table 3.2.

Technically speaking, a robot is a mechanism able to move by displacing joints. The kinematic chain in a robot manipulator can have mathematical modeling tools such as

Table 3.2. *Technical characteristics of ABB YuMi IRB14000*

| Feature | Value |
|--------------------------------------|-------------------------|
| Total number of axes per arm | 7 |
| Number of arms | 2 |
| Protection index | IP30 |
| Reach per arm (Max) | 559 mm |
| Payload per arm (Max) | 500 g |
| Tool Center Point velocity (Max) | 1.5 m s^{-1} |
| Tool Center Point acceleration (Max) | 11 m s^{-2} |
| Position repeatability | 0.2 mm |
| Total weight | 38 kg |
| Communication interface | 100/10 Base-TX Ethernet |

**Figure 3.4.** *ABB YuMi IRB14000, cobot used for experimentation [1]*

Denavit–Hartenberg parameters, direct and inverse kinematics among others. These mathematical tools allow to find specific displacement values for each joint (rotational or linear) to lead the movements of the robot to a specific place. Usually the displacement is carried by a servo motor connected to a electronic drive. This last device is able to control motion parameters on the servo motor as velocity, acceleration, position etc. by producing customized electrical power supply signals. An interface to setup motor drives can be an interactive panel (front panel or handheld) or a physical communication port to provide data accessible through a computer software. For commercial models of robots, a centralized controller, is able to govern all the motor drives, based on computational processing of mathematical and logical models, as mentioned before. The high level control parameters and custom execution sequences are defined by the user and can be monitored and changed through a specific computer software using parameter lists and

specific programming languages (usually proprietary licensed). Robot controllers usually exchange information with external software through standard communication protocols by serial interface.

To program custom execution sequences on robots, each order can be given as a command of joint rotations (joint space) or 3D-coordinates desired to reach (Cartesian space), this last are usually called targets. By giving coordinates in Cartesian space, the reference frame should be defined. Usually the frame in use can be referenced to global, base, work piece and TCP spatial locations. The user can indicate a Cartesian space destination, described as offset coordinates referenced to the most convenient frame for each target.

Contemporary software tools for robot programming allow the developer to have useful features as a 3D model of the robot and its environment, a virtual controller and the possibility of testing its execution sequence code virtually, before download it to the robot controller. ABB robots can be programmed through its proprietary software *RobotStudio*. This software allows to access and program robot parameters in virtual and on-line mode. The programming language for this software is called *RAPID* [55].

Conventionally, the creation of an automated task for robots, starts by developing the execution sequence required in the process by defining a set of targets based on virtual recreations from 3D models. Then, a sequence of displacements through targets is coded, creating paths. The execution of multiple paths is controlled by triggers, such as inputs from a HMI interface or process sensors. After this, the program is uploaded to the robot controller. Lately, the targets are redefined one by one, based on real world Cartesian space coordinates. The redefinition of targets, also called teaching, is performed by site engineers leading the robot to a desired pose through a teaching pendant (hand-held interface). Finally, the execution sequence is played and the robot controller resolve all the displacements required by each joint to navigate through targets and complete successfully each path on the process.

The methodology to create an automated task for robots, might differ for the particular case of cobots, thanks to the direct interface programming feature explained on Chapter2 (lead through). While the methodology for conventional robots aforementioned require a redefinition of targets on site by leading the manipulator using a teaching pendant, cobots can be guided to a desired target by direct interaction with the manipulator. In practice, this makes quicker and easier to drive the robot on site through each target planned. During the implementation of this methodology it was found that all targets created in a virtual 3D model, based on exact alignment and edge locations of the virtual case faces, were solved and executed by the robot controller using manipulator configurations that might result on discomfort for the human operator. The cause of discomfort is supported by the fact that the manipulator body obstructs the free visibility and access to the work pieces and even worse, paths just cannot be executed, because each pose is not process aware. Figure 3.5 presents all the configurations solved by the virtual robot controller to reach one of the work pieces. The elbow of the manipulator is standing out, towards the

human operator area. Figure 3.6 presents the error logged by the controller, while executing a simple path between virtual case faces, making impossible to complete even half of it. This evidence demanded to reformulate the methodology and rethink an additional stage of iterative pose sculpt and path execution tests. First a virtual approach was followed, but sculpting a pose in a 3D model can be more time consuming than just doing it with the physical manipulator on site. Just for testing, the same path presented on Figure 3.7 was re designed by sculpting a pose comfortable for the operator, then the controller could solve and execute the path successfully without any error.

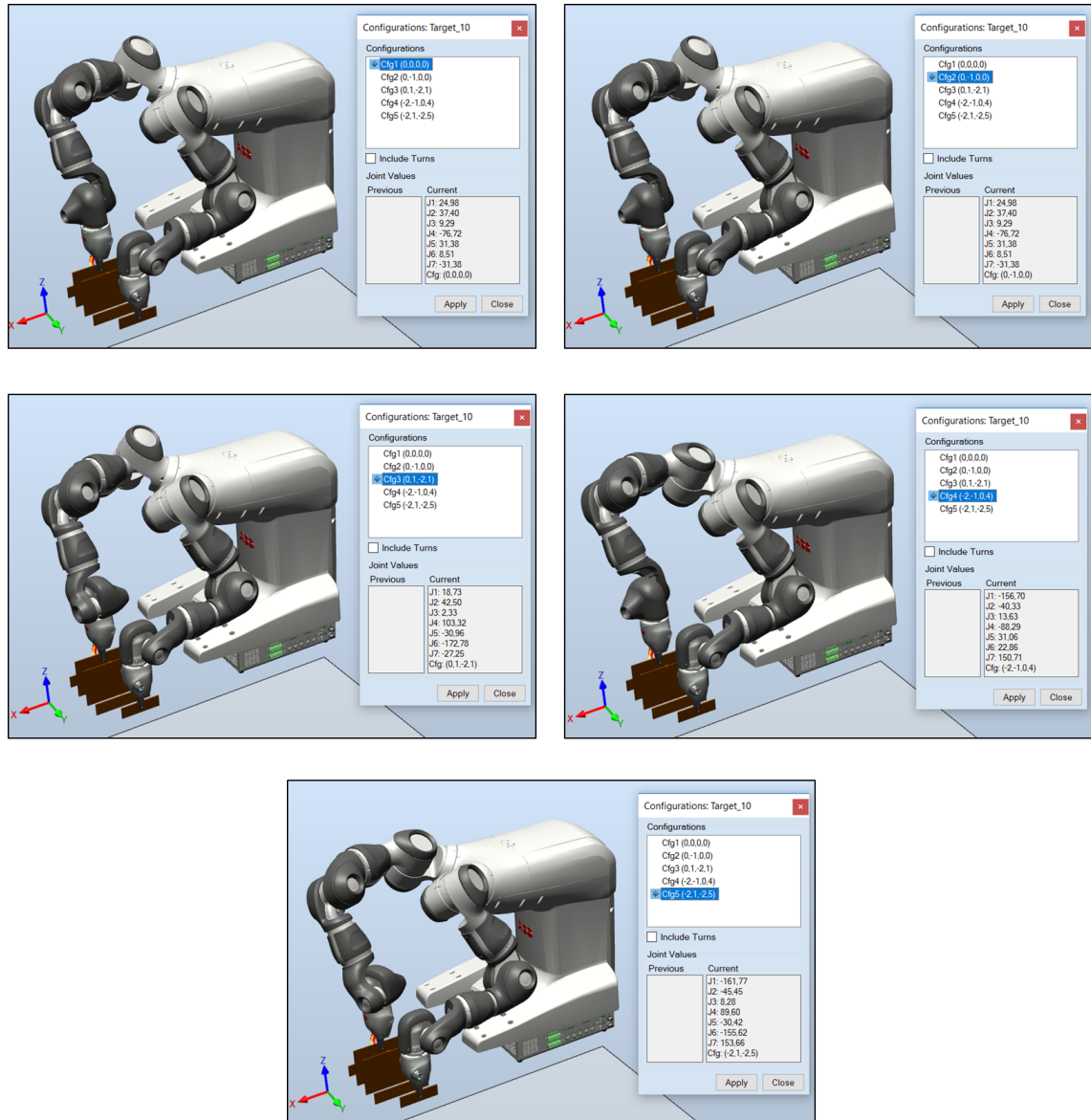


Figure 3.5. Virtual manipulator configurations self-generated by YuMi to reach a work piece

By adapting or "sculpt" the poses of the cobot to coexist with a human during the process execution, is noticed that human-comfort factor start playing a relevant role on the kinematic planing of cobots. Characteristics as avoid visual field obstruction, perform human-like linear movements and keep the cobot arm body as far as possible of the

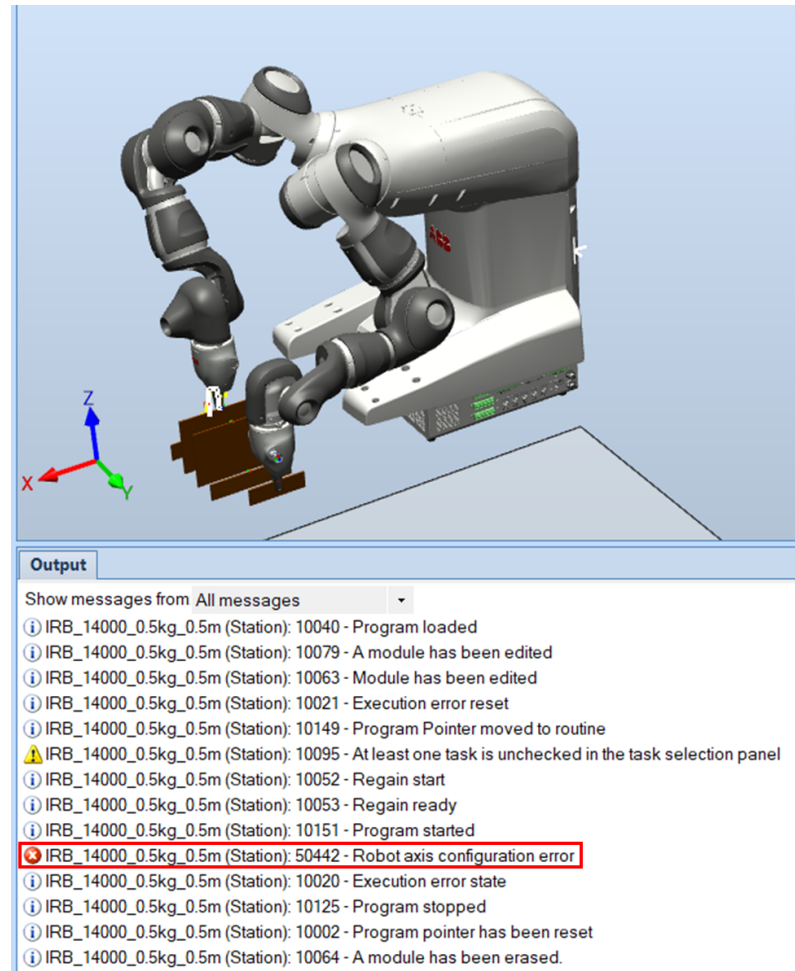


Figure 3.6. Error on automatic path generation by YuMi

human operator, can contribute to a more comfortable user experience. The proposed methodology of pose sculpting by the lead through feature, makes possible to influence the robot controller through convenient configurations for the process and the human operator. Therefore, to start teaching the cobot by direct interface, the site engineer can sculpt manually the cobot poses based on human-like movements, then, teach the targets and observe the automatic kinematic execution solved and executed by the cobot controller.

Paths following end-to-end process targets are used often to simplify the kinematic execution sequence. This situation might cause errors during the trial path executions as singularities, undesired trajectories, and zones out of reach. To solve this inconvenience, it will be necessary to add intermediate targets as via points, until the robot execute automatically effective displacements, comfortable for the human operator and effective for the assembly process. The methodology presented will be named by the author in this work as Enhancement of Cobot Kinematic Behavior (ECKB). Detailed instructions are summarized by a flow chart presented on Figure 3.8.

By testing the ECKB in tab-slot matching tasks for the modular case assembly process, it was found that accuracy and precision was poor, since sequential assembly processes

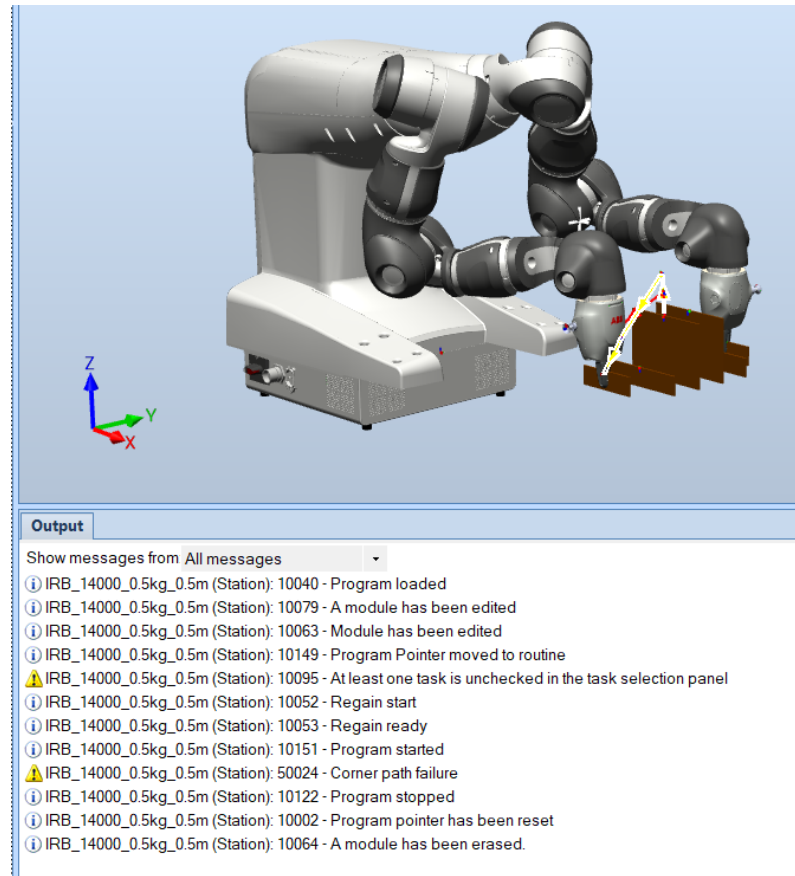


Figure 3.7. Correct path generation by YuMi aided by previous pose sculpt

guided by predefined targets are susceptible of cumulative grasping error. This phenomena comes from occasional drifting of the work piece grasped by the robot, effect of involuntary contact between the work pieces and the site engineer, while performing pose sculpting or target teaching.

Since this experiment does not contemplate any external vision assisted technology to enhance the cobot accuracy while fitting the case faces, and the poor accuracy by cumulative grasping error caused unacceptable low precision on the assembly task; it was necessary to formulate an additional stage to ECKB to complete successfully the teaching methodology for the cobot. Instead of reaching directly the face-matching target by pose sculpting, the site engineer can apply ECKB to a near-face-matching position, then teach it as a dummy target and then drive the manipulator to pick the face to be matched. Afterwards, jog the manipulators to the near-face-matching target and then drive it carefully to a face-matching target by using the teaching pendant on linear jogging mode with reduced speed. This allows to ensure a correct trajectory by ECKB, match accurately all the tab-slot sets and record the exact face-matching position. Finally, from the face-matching target an additional target should be added, by adding a linear perpendicular translation (offset) to the face-matching target. This additional target will act as a pre-matching position for the cobot, reachable quickly before start a linear approach towards the face-matching target.

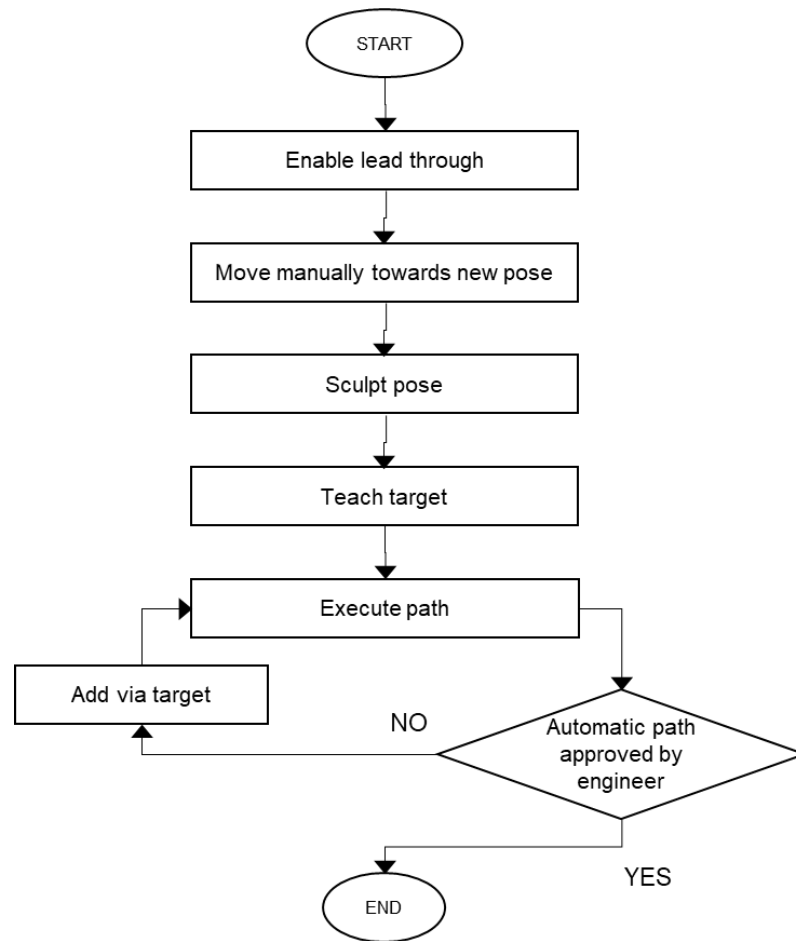


Figure 3.8. *Enhancement of Cobot Kinematic Behavior (ECKB). Flow chart*

Finally, if during execution tests the path generated by the robot controller has any error, collision or human comfort concern, is possible to include additional via points follow the ECKB methodology. Internal factors as velocity and corner avoidance might affect the fluent displacement of the robot. For that reason it was determined to execute any path away of the face-matching targets using v200 (TCP speed: 200mm s^{-1} , orientation speed: $500^{\circ} \text{s}^{-1}$). For face-matching related trajectories is used v50 (TCP speed: 200mm s^{-1}) The methodology explained to teach how to match faces is named by the author as Reverse-Matching Assembly Technique (RMAT). It prevents any contact with the work piece during the delicate tab-slot matching manipulation.

After implementing a combination of ECKB and RMAT, accuracy and precision on the assembly process was appropriate, however, during assembly the operator can correct easily any minimal grasping error, without triggering any torque alarm on the cobot. Figure 3.9 presents a flow chart describing the complete teaching method to match each face during the assembly process. As explained, it includes a synergy of ECKB and RMAT. The teaching method was repeated for each face, obtaining a set of targets to be called by the execution sequence.

The hierarchy of dual arm manipulation established by C. Smith in [48] defines the assembly process carried for YuMi in this project as goal-coordinated manipulation, since

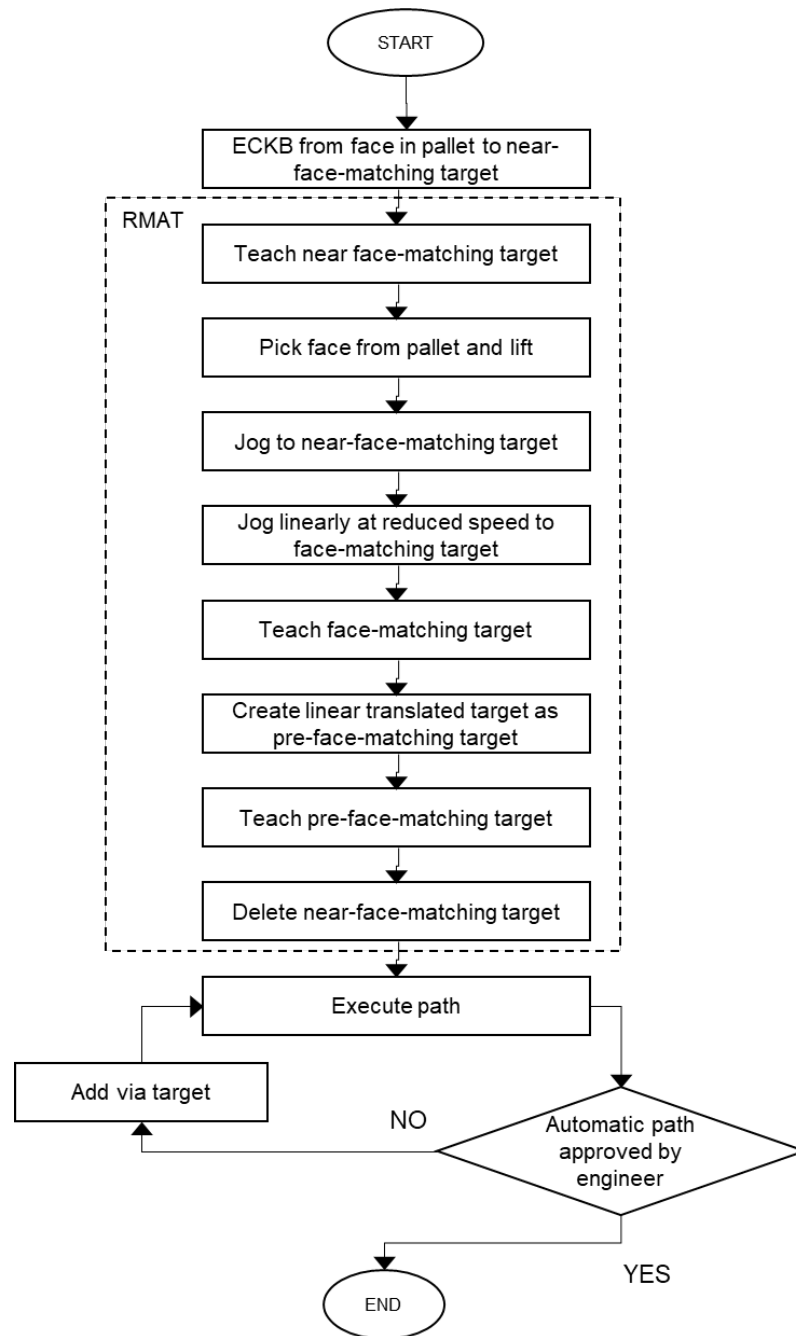


Figure 3.9. Teaching method for face matching (ECKB and RMAT). Flow chart

both arms carry parts independently, but work together to achieve the same goal (match faces). In this case, RAPID language supports dual arm manipulation in a shifted and simultaneous approach. Despite every manipulator connected to the same controller has a separate run-time, homonym variables on each run-time type *syncident* and commands as *SyncMoveOn* / *SyncMoveOff* or *WaitSyncTask* make possible to execute simultaneous and shifted dual arm manipulation.

Having all the teaching methodology analysis, is possible to complete the coding process in YuMi. For that, trajectory parameters as moving type (linear or not-linear), speed, corner zones and coordinated synchronized move instructions, should be defined for

each path. The author listed some guidelines as common reference to code trajectories and actions executed by YuMi:

- The grasping force of the griper should be 10 to 15 N. This is formulated from experimental observation.
- The paths to extract the faces from the pallet should be slow, linear and avoid corner zones
- The paths to transport the faces to a pre-face-matching target can be quick and not necessarily linear
- The paths to and from every face-matching target should be slow, linear and avoid corner zones
- Dual arm manipulation on any trajectory not relate to the face-matching target can be shifted
- Dual arm manipulation to and from face-matching target should be simultaneous

The final version of the code implemented in RAPID is presented on the Appendixes A.1 and A.2 for the right and left manipulator respectively. Additional features on lines 58 to 154 of A.1, as functions for WebSockets and integrated vision classification will be explained on Sections 4.4, 4.5 and 4.3.

3.3 Integration methods for a human-robot collaborative workstation to FASTory

The ability to interoperate is an essential feature of the proposed human-robot collaborative assembly workstation. As mentioned before, FASTory has full support for Ethernet technologies, since its orchestration functions are carried by an OKD-MES, based on web APIs to invoke service calls. The services invoked are queued and executed by one or many RTUs, installed located on every cell.

This project will provide a framework based in SOA, to enable the integration of agents involved in the collaborative process such as Human, YuMi, AMIR, conveyors and the general process orchestrator. It is planned to enable the interaction between the agents mentioned, by using Ethernet technologies and web application architecture. The approach in this project reviews general aspects initially, then, details required for implementing effectively the solution on each agent will be addressed.

Table 3.3 presents a technology assessment for each agent on the collaborative workstation environment. As seen on the network interface columns, it is required to bridge the wired and wireless Ethernet connections. Meanwhile, the application interface assessment present all devices compatible with TCP/IP. Some of them use RESTful Web Services (RWS), based on HTTP requests marked with *R* and others use Sockets [64], marked with *S*. In contrast, local support for ROS is not present on all agents, since is not

available for the Inico S1000 RTU units, controlling the conveyors transporting the case parts. Even is feasible to develop a ROS support package for Inico S1000, it is not part of the scope of this work and there is already the TCP-IP alternative, which is exempt of any additional effort. Then, At network interface level, compatibility does not represent a technological challenge, since it just require any basic model of wireless access point to bridge the physical connection media, but at application level web services is fully compatible on the environment, however operate successfully RWS and Sockets might require special software tools to complete a reliable implementation.

Table 3.3. *General assessment of communication technologies agents*

| Agent | Controller | Network interface | | Application interface | |
|-----------------|-------------|-----------------------|-----------------------------|-----------------------|----------|
| | | Ethernet 100BASE-T | Wireless IEEE 802.11n | TCP/IP | ROS [62] |
| Conveyors | Inico S1000 | ✓ | | ✓(R) | |
| Dual robot cell | ABB IRC5 | ✓ | | ✓(S) | ✓ |
| AMIR | MIR-100 | | ✓ | ✓(R) | ✓ |
| YuMi | ABB IRC5 | ✓ | | ✓(S) | ✓ |
| Orchestrator | Server | ✓ | ✓ | ✓(R+S) | ✓ |
| HMI | Tablet | | ✓ | ✓(R+S) | ✓ |

From the analysis of Table 3.3, integration technologies were selected and sketched on Figure 3.10. Every line represents an Ethernet link, the ones with a black waved symbol represent wireless links.

From a functional perspective of integration, the human is expected to control the collaborative process flow, since the cognitive capabilities of the operator are critical for HRC. A touchscreen tablet can be used as HMI, which dynamically can provide buttons and animations that guide the operator through the process and allow him to control the process flow by enabling specific triggers on each stage of the process. The HMI is expected to be available online, accessible from a web browser and should include animated buttons and an illustrative videos of the process execution. Section 4 provides detailed information about the design of the HMI web application.

This chapter illustrated the context for the projected human-robot collaborative workstation and FAST-Lab. Then, the demonstrative process to extend the functionality of FASTory was selected and specified. This process allow to assembly a special modular case, to pack the products provided by the current implementation of FASTory. Finally, the integration methods where defined to enable interaction between the cobot, a human, conveyor belts, an AMIR and a process orchestrator. The next chapter will define detailed engineering specifications and propose an experimental test method to evaluate the results of implementing the new workstation.

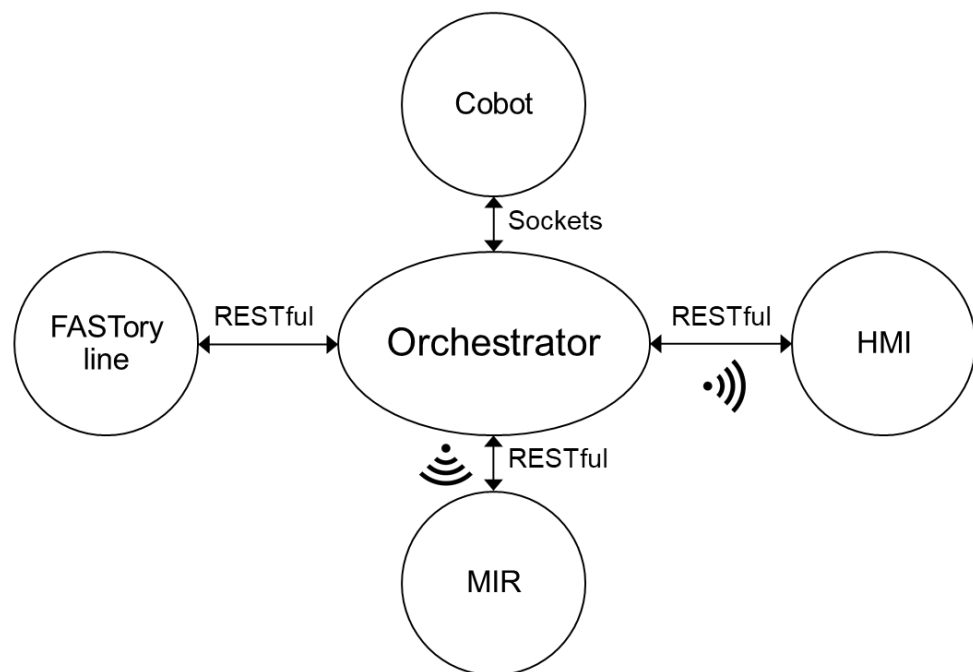


Figure 3.10. Sketch of integration for agents in the modular production line

4 IMPLEMENTATION AND RESULTS

The scope of research for this project comprises the implementation of a human-robot collaborative workstation integrated to several adjacent automated systems. To achieve this purpose, technological tools selected must interoperate as a complete scalable solution. By providing detail engineering specifications for producing a first functional prototype, this work will allow to demonstrate a real scenario of HRC in FAST-Lab.

FAST-Lab environment has strong influence from previous research works on SOA presented by B. Ramis on [41] and V. Herrera on [25]. Despite this precedent about the integration of automated agents in FAST-Lab and some principles relying on Ethernet technologies and SOA given on Chapter 3, it is necessary to specify how the assembly process proposed needs to be programmed on the cobot and what tools will enable all the agents to interact on the new environment.

This section present a compilation of observations and experiments carried in FAST-Lab to verify the methods proposed in Chapter 3. All the technical specifications (detailed engineering) necessary to provide effective operational methods for the new collaborative workstation and its new interconnected environment will be addressed. Initially, a general context will be illustrated by presenting information about FASTory, the modular robotized production line at FAST-Lab serving as research environment. Then, the guidelines to implement a workstation able to reproduce the process selected will be provided. Afterwards, specific information will provide details about the cooperative product classification, the backend integration, HMI and how to prepare the adjacent devices on the production line. Finally a test method will be formulated and the results of the functionality test will be presented.

4.1 The new FAST-Lab

One of the most important research topics in FAST-Lab is the application of the latest technology to enable the coordinated operation of diverse industrial manufacturing devices, using industrial communication standards (interoperability). The purpose behind incorporate assorted types of robots in FAST-Lab is to have a research facility where is possible to implement interoperability scenarios into demonstrative industrial processes. FAST-Lab offers the opportunity to demonstrate the capabilities of fully automated industrial environments, where several combinations of robots and humans are feasible to

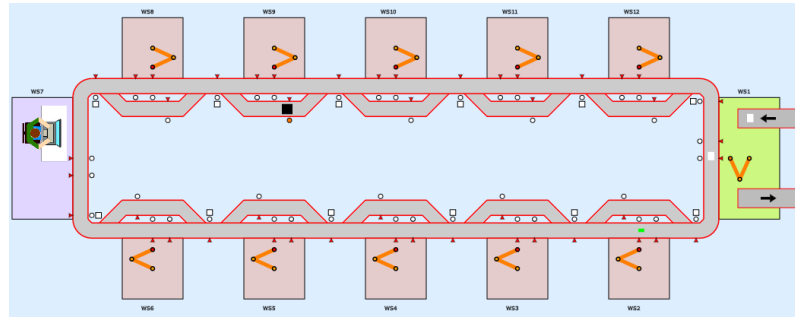


Figure 4.1. Initial structure of FASTory implemented in FAST-Lab [18]

implement.

At early formulation stages, formalize new ideas to extent the features available in FAST-Lab was included on the scope of this work. The ideas were provided collectively by the research staff at FAST, but later, it was decided to focus efforts on specify in more detail the human-robot collaborative assembly workstation. By compiling those ideas, it was planned to provide new environments for research projects and collaboration between academy and industry. This precedent, established the existent features in FAST-Lab, as technological context and pillars for the final formulation of this project. Former H2020 projects hosted in FAST as C2NET or eScop, provided strong influence on the transformation towards a renovated laboratory.

The conventional process in FASTory, ran around a closed loop of unidirectional conveyors, moving pallets through robotized cells, as shown on Figure 4.1. On each cell, a manipulator can draw telephone components over a paper sheet carried by a pallet. Each robot cell has a bypass conveyor that enables the transit of pallets while the robot area is in use. The control and orchestration functions in FASTory are carried by an Open Knowledge-Driven Manufacturing Executing System (OKD-MES), implemented from the eScop project and documented by several authors on [37], [26], [36] and [19]. Figure 4.1 presents the aforementioned structure of FASTory, based on robot cells.

Remote Terminal Units (RTU), compatible with RestFul web services, work as IO interface for the OKD-MES on each cell. Additionally, these devices are equipped with digital IO boards and have capabilities to execute low level control logic sequences, implemented on a customized version of Structured Text (ST) programming language from IEC 61131-3. Every RTU is interconnected by a common Ethernet wired network, concentrated in a managed Ethernet switch. All RTUs are implemented on FASTory using INICO S1000 units and the system is based on Service-Oriented Architecture (SOA).

The execution of every low level control logic sequence by the INICO S1000, is based on its ability as RTU to queue service calls, invoked using web APIs, triggered from the OKD-MES or any other device TCP/IP enabled in the network. Web APIs abstract the services available on each cell and conveyor, simplifying the way of invoking them and providing multi-platform compatibility as a RestFul web services implementation, requested through HTTP methods. Also, the RTU features allow to subscribe to predefined events on each

cell, as sensor changes or updates on status of the robot.

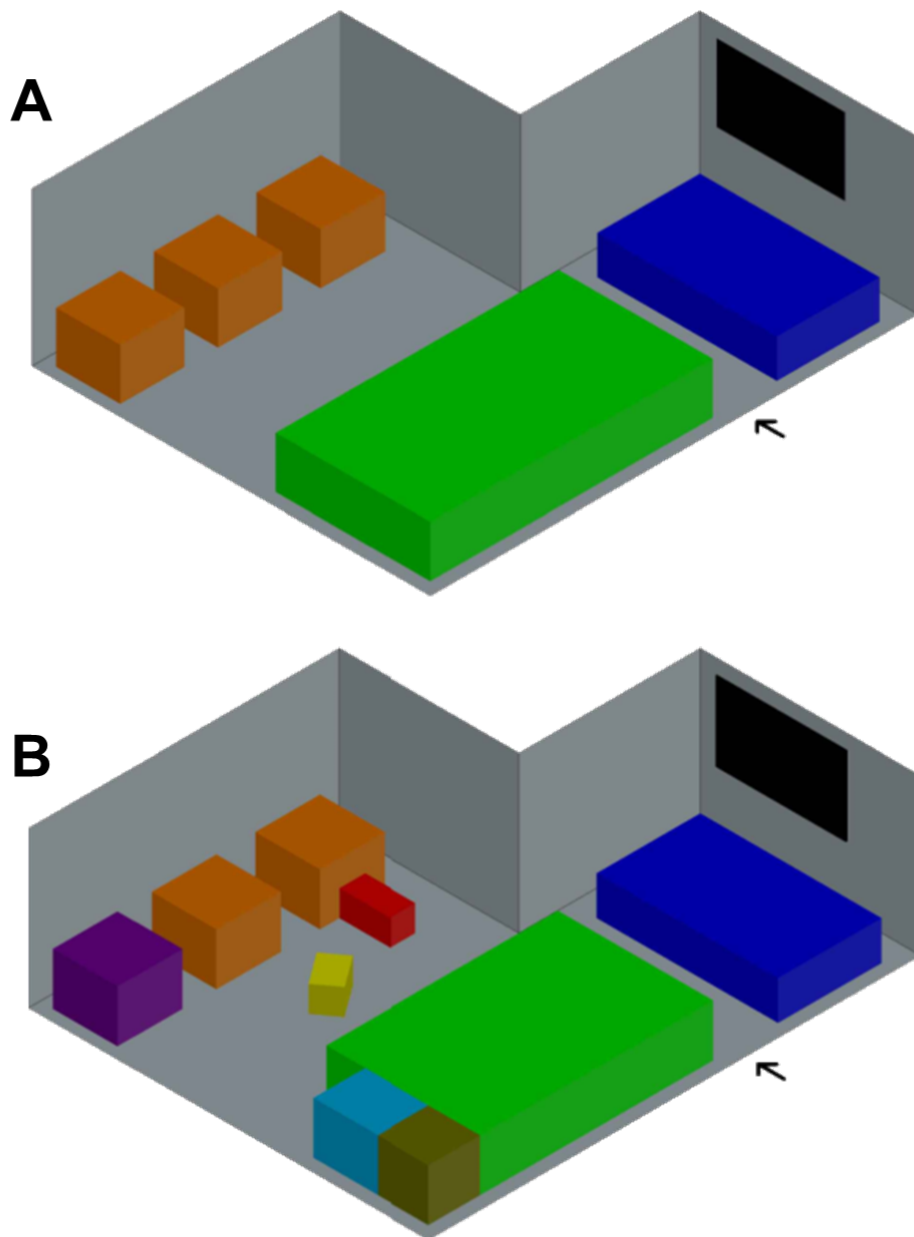
The new consolidated process that extends the functionality of FASTory, starts by feeding a paper sheet of 105 mm by 150 mm as raw material into a pallet. Raw material and pallets are stored in the storage buffer cell, where supplies are handled by an industrial dual arm robot (SDA-10F) embedded on the material handling cell. Then, the pallet is guided across a closed loop conveyor system, running through 11 workstations, where is possible to order 729 product variants out of 3 components, with 9 different characteristics (3 models in 3 possible colors). After production, the pallet returns to the initial cell, where the SDA-10F robot picks the product and loads it into a transport basket. The OKD-MES system checks if the order requested is completed or the basket is fully loaded. In any of the last two cases it processes a service requirement for an Autonomous Mobile Industrial Robot (AMIR), to transport the product basket to the next stage. The dual arm robot loads the product basket and materials for the shipping box on the AMR. Then, the orchestrator requests the AMR to continue its way to the next stage. The AMR navigates through the laboratory and stops beside the human-robot collaborative workstation. Additionally, in the future the AMR can transport materials required for the FESTO MPS-500 Line or the parallel robot cell.

Having this context is critical to formulate a demonstrative human-robot collaborative assembly process, that complement the process aforementioned by compatible technological means. Even so, the modifications presented will foster the diversification on robotic scenarios and interoperability aspects on FAST-Lab research projects, it will require the coordinated completion of several individual renovation projects as this one. Figure 4.2 presents a conceptual 3D model of both, the transformation projected and the status on 2018 of the laboratory.

4.2 Implementation of a human-robot collaborative workstation, suitable for the modular-case assembly process

The area of work where human and YuMi will execute the process selected, requires to be flat and attached to the main pedestal structure. This will allow to have a planar and constant reference frame for the cobot. Other specifications will be analyzed and defined then.

The mechanical design of the human-robot collaborative workstation, needs to comply health and safety requirements, but the literature review done, could not find specific standards on ergonomics for human-robot collaborative workstations. Additionally, there are technical requirements to comply in the FASTory process, such as range of action (reach) of all the robots involved. Considering essential the ergonomic recommendations for work surface in conventional computer workstations from section 4.1.6.1. of [40], it was decided to establish a height of 820 mm for the flat surface on the collaborative



| LEGEND | |
|--|--|
| ■ FASTory line | ■ Parallel robot cell (Omron Quattro) |
| ■ Storage buffer cell | ■ Dual robot cells (ABB IRB140) |
| ■ Material handling cell (Yaskawa SDA-10F) | ■ Autonomous mobile industrial robot (MIR-100) |
| ■ FESTO MPS500 line | ■ Human-robot collaborative work station (ABB YuMi) |
| ■ HMI video wall | ➔ Entrance door (Ri208) |

Figure 4.2. Conceptual 3D model of FAST-Lab: (A) Late 2018 (B) Transformation projected

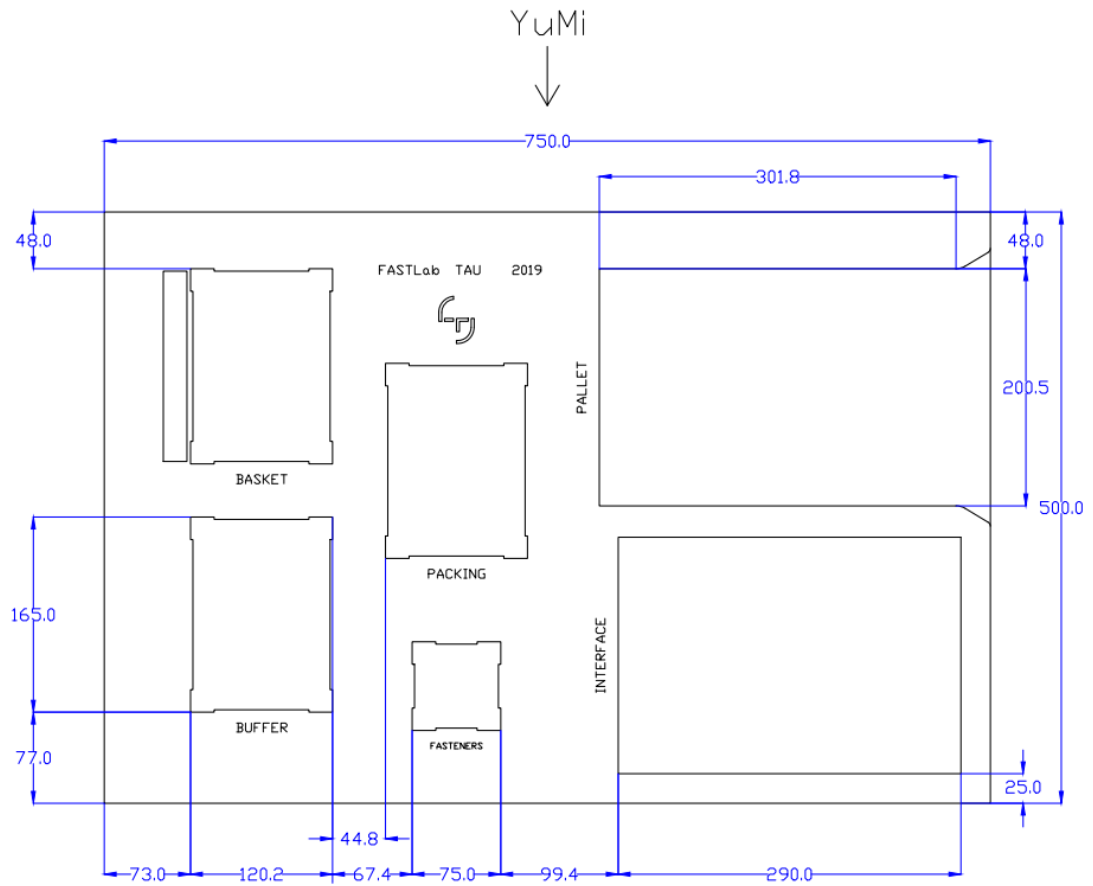


Figure 4.3. *Slotted surface distribution design for the collaborative work space*

workstation. This dictates the height of the work surface reachable for the human and robot on the collaborative work space. As well as the AMIR transport frame and feeding conveyors for the dual robot cell.

The work space area contain the pallet carrying the case parts, a flat HMI screen, a box of fasteners, two baskets (for new products and buffering), the blower module for correcting single-sheet paper grasping of YuMi and the case assembled in the process. Every component mentioned is expected to have a slot, cut in a flat board of 3mm thickness. Figure 4.3 presents the distribution designed for the work space, for laser cut production. The distribution was made over a 750mm by 500mm surface, that gently extends the front and depth dimensions of the workstation. Making it easy to go through standard door frames.

The internal work space distribution aims to locate the components conveniently to avoid collisions between both arms of YuMi. For that reason, the pallet carrying the faces for the case, has a distribution as shown in Figure 4.4. The bigger faces are located in the center, preventing possible obstruction for YuMi, while the smaller are on the lateral outer slots.

During laboratory tests performed to YuMi, for grasping and manipulating case prototypes, the default finger model included in the smart gripper, presented an ineffective

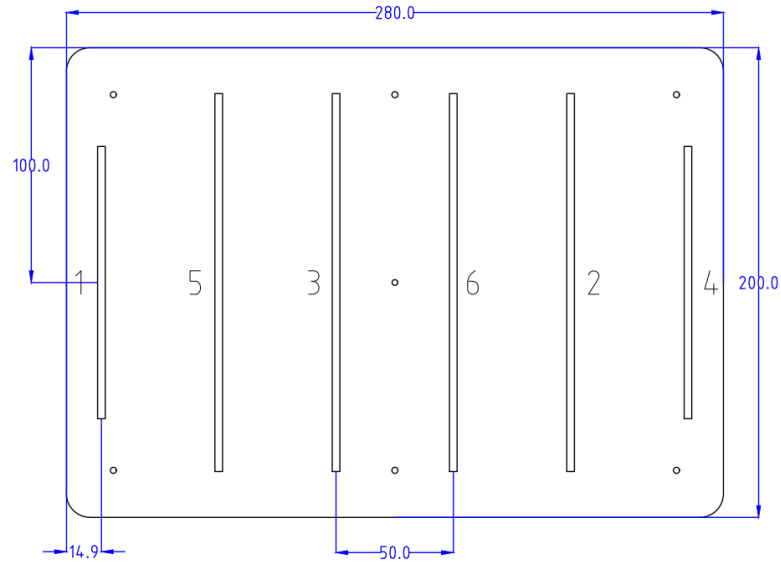


Figure 4.4. *Slotted surface design of the pallet for the modular case faces*

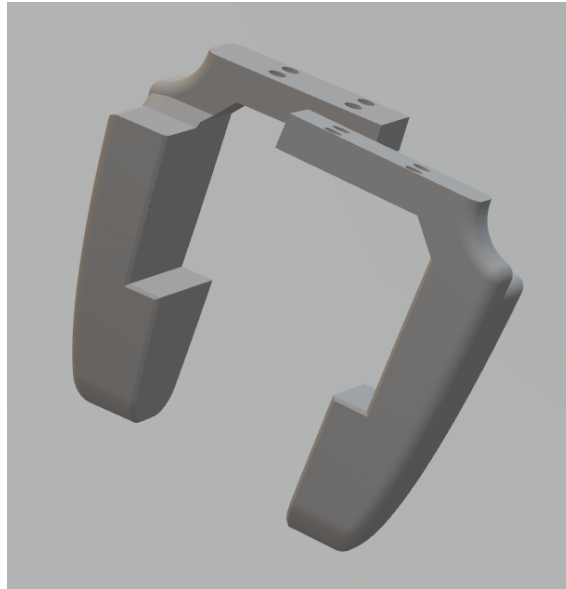


Figure 4.5. *3D model of the customized design for YuMi smart gripper fingers*

behavior holding and transporting flat smooth surfaces (case faces). As suggested on the product specification manual for YuMi (3HAC052982 PS IRB 14000-en) [40], is possible to design customized fingers, based on pre-defined dimensions. Then, a finger model designed by W. Mohammed in FAST-Lab was produced, tested and enhanced by the author to fulfill the requirements in the human-robot collaborative process. The customized fingers designed locally and 3D printed on engineering grade thermoplastic (Onyx) with carbon fiber reinforcement, consists of a tip-extruded finger, with rubber padding for better friction over the contact area [48]. The final design is presented on Figure 4.5.

Finally, it was necessary to provide specifications to modify the pedestal structure for the cobot. This will define the materials required for conditioning the pedestal and extend

its functionality by installing the collaborative work space presented previously on this chapter. The material selected for the surface was laminated wood of 25mm thickness, which is the same used in all the computer workstation of the laboratory. The design is based on industrial aluminium profiles produced by *Item*. The bill of materials is presented on Table 4.1 and the dimensional specifications are presented on Figures 4.6 and 4.7.

Table 4.1. Bill of materials for collaborative work space design.

| Quantity | Part number | Name | Remarks |
|----------|-------------|--|-----------|
| 6 | 0.0.265.31 | Hinge 8 40x40, heavy-duty | - |
| 4 | 0.0.625.23 | Angle Bracket Set 10 50x50 | - |
| 2 | 0.0.681.81 | Table Top 25 plastic coated, grey, similar to RAL 7035 | 750x700mm |
| 8 | 0.0.026.18 | T-Slot Nut 8 St M8, bright zinc-plated | - |
| 8 | 8.0.000.19 | Button-Head Screw M8x16, bright zinc-plated | - |
| 6 | 0.0.265.31 | Profile 8 40x40, natural | 3m |

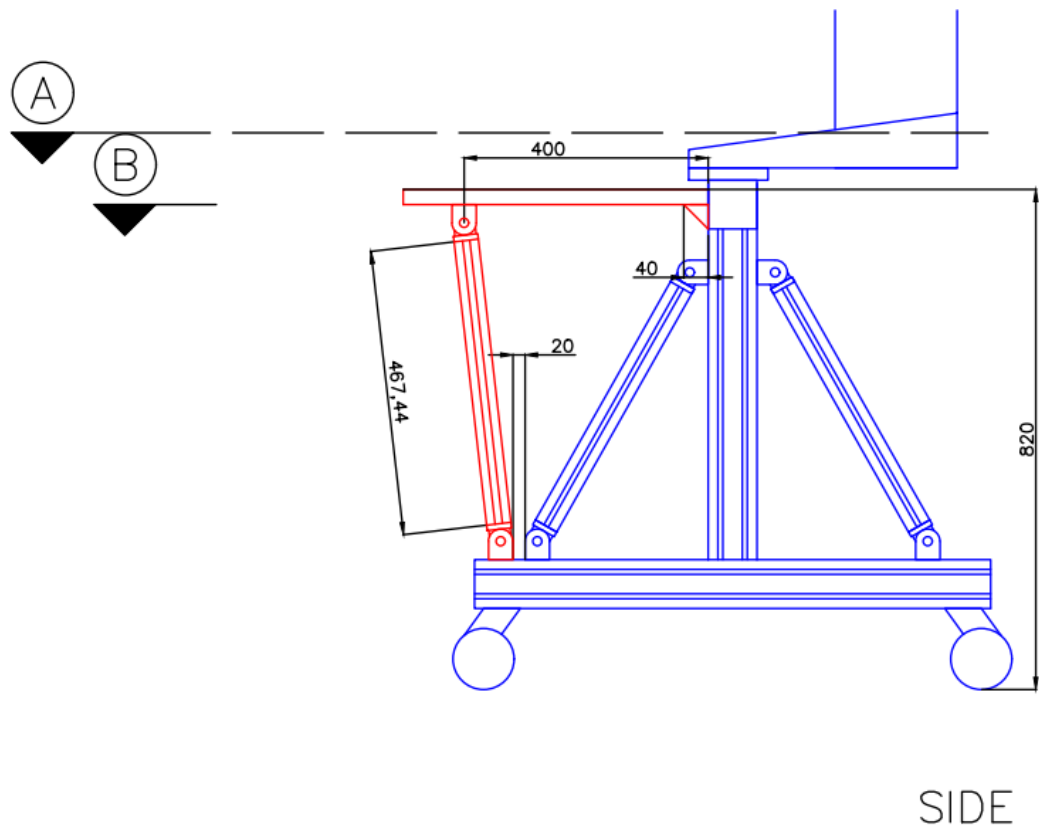


Figure 4.6. Side view of pedestal for YuMi

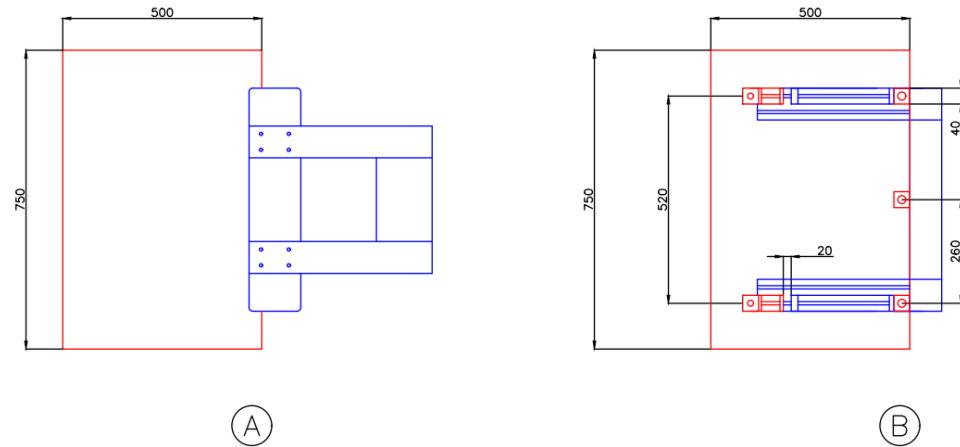


Figure 4.7. Plant cut views of pedestal for YuMi

4.3 Cooperative product classification

ABB YuMi is compatible with a multi functional smart gripper model IRB14000 provided by the same brand. The modules on the gripper can include dual finger servo grasping, vision (smart camera) and vacuum grasping (suction cup). The specific YuMi unit implemented on the human-robot collaborative workstation prototype, include smart grippers on both manipulators. A servo-vision-vacuum smart gripper on the right side, while a servo-vacuum on the left. In this case, the smart gripper in use will be only the one installed in the right arm of YuMi. Each module on the gripper can be used on run-time by commands programmed on the RAPID scripts of the host manipulator (see lines 77-114 of A.1). This devices make possible to extend the scope of the workstation, to include a shifted process (cooperative), such as product classification, beside the collaborative assembly process. This gives realism at big scale to the general process of FASTory and allow future experimentation for both, collaborative and cooperative tasks.

As explained in Chapter 3, the current structure of FASTory line, produce paper printouts of telephones. A product classification can be carried out by using the vision and vacuum modules. First, the manipulator will reach a preset target above the product basket, where the integrated camera can take a black and white snapshot of the top product. Then the smart camera will compare it internally with pre-trained images on the ABB-Cognex PatMax tool. The vision module runs a verification evaluating a likelihood score (Accept threshold), to validate if the shape on the snapshot corresponds to any phone model known. The training images used for demonstrative purposes correspond to 3 types of smartphones and the box bottom. In the near future FASTory can change the printouts designs, in that case, a sample of the new printouts is enough to train again the smart camera identification function. Figure 4.8 presents the programming interface for the vision system on RobotStudio 32-bit version. Table 4.2 presents the settings used for ABB-Cognex PatMaxm, including acceptance . Figure 4.9 presents the graphic models of smartphones used as demonstrative printouts.

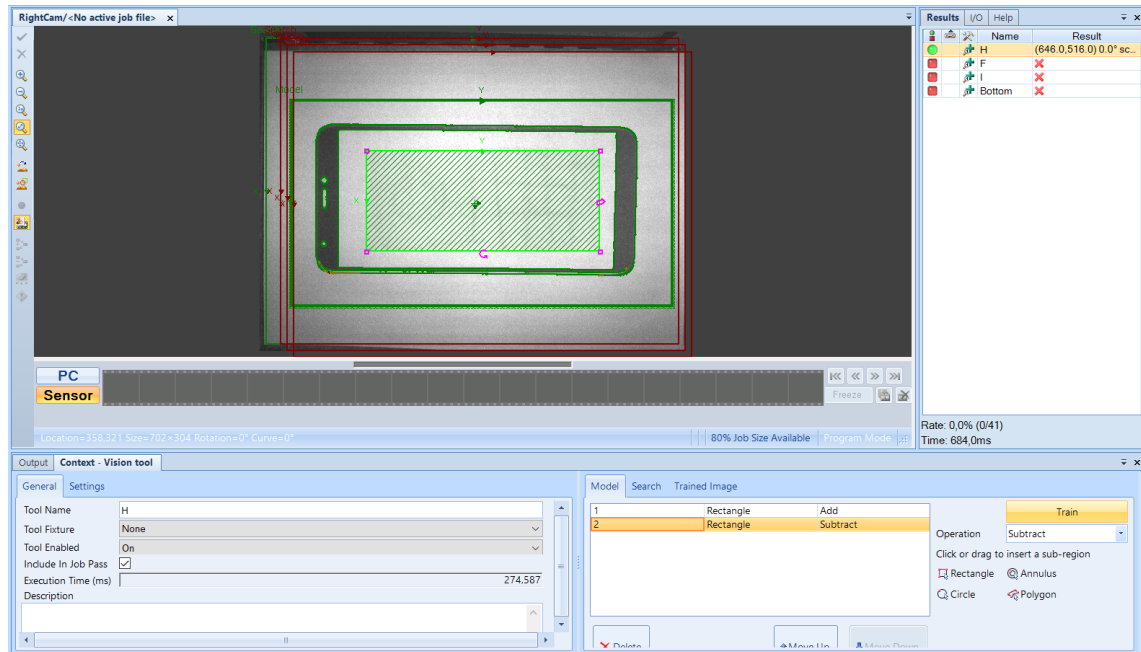


Figure 4.8. User interface of vision system component in RobotStudio

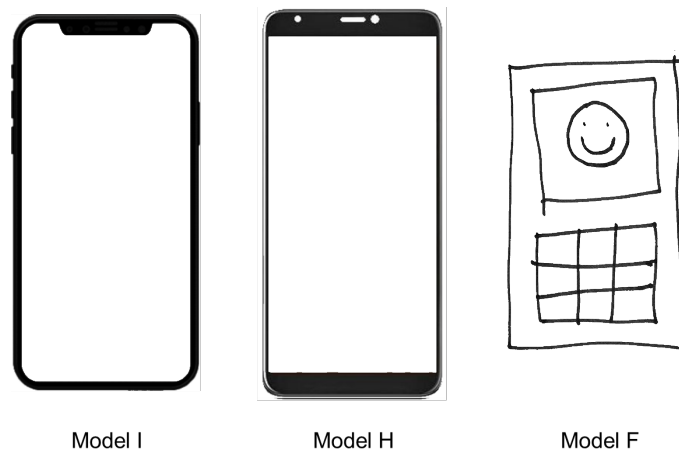


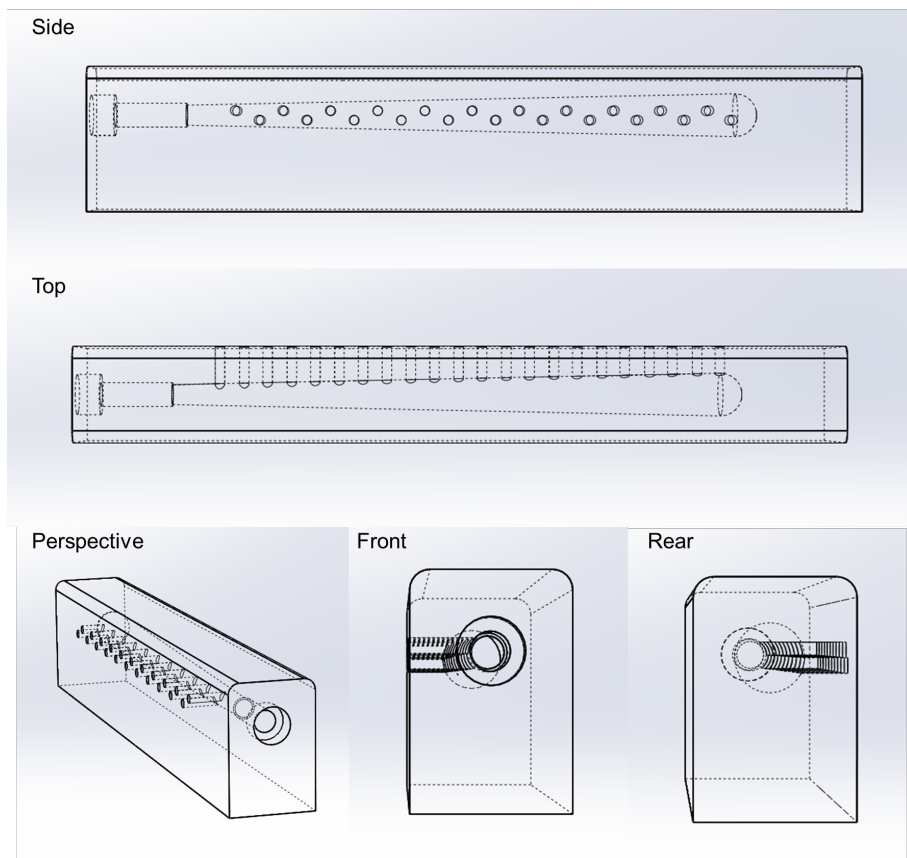
Figure 4.9. Demonstrative graphic models for cooperative classification

Later, YuMi might grasp the top paper sheet by using the vacuum module to classify it. The classification is based in a countdown of the amount of units per model, explicitly requested by the EnableCollab web service from the OKD-MES. The number of iterations for the classification routine is reported to YuMi by the orchestrator when the product basket is taken from the the AMIR (PickBasket web service). The AMIR must be dispatched previously from FASTory, carrying products from the material handling cell to the collaborative workstation. The classification process leads YuMi to deposit the exact amount of products ordered by the orchestrator into the open modular case (collaboratively assembled), while any additional, defective or non-recognized model, will go to a neighbor buffer basket to be stored lately.

During functional tests, more than one sheet was occasionally grasped at the time by YuMi. These additional sheets were dragged up by effect of mechanical deformation of

Table 4.2. Vision system at YuMi. PatMax pattern settings

| Parameter | Value |
|--------------------|-----------------|
| Tool name (model) | I, H, F, Bottom |
| Accept threshold | 85 |
| Contrast threshold | 10 |
| Rotation tolerance | 180 |
| Scale tolerance | 0 |
| Exposure | 0.9ms |
| Number of rows | 960 |
| Light intensity | 255 |

**Figure 4.10.** Hidden line view of air nozzle for air blast paper separation method

the top sheet or electrostatic charges among them. This phenomena affects critically the countdown for classification. To overcome this issue, it was necessary to add a controlled horizontal air blast, to separate the paper under the top sheet. Additionally it was necessary to reduce the speed of the manipulator while extracting the sheet out of the basket, this allowed to perform effectively the air blast method, without dropping the top sheet. The model of the air nozzle was produced by S. Vasudevan and the prototype was 3D printed locally on medium density Polyactic Acid (PLA). The design is illustrated on Figure 4.10. The airflow control was implemented by a 24vdc pneumatic electro valve, connected to the digital output 7 (DO7).

4.4 Backend integration

As presented on Figure 3.10, interoperation for all agents will rely on a bridged Ethernet connection (wired and wireless). These lower level communication links enable the agents to configure a SOA. Therefore, any agent connected to the TCP/IP network can invoke a set of services by abstracted HTTP methods or peer-to-peer sessions (Sockets), also known as API.

The workflow for invoking a service begins by an API call from an agent (client) referred to as the service provider agent (server). As soon as the server receives the API call, it executes concrete local actions defined in scripts for the specific API invoked. The actions carried locally by server by an API call depend on the local functionalities, e.g. an RTU activates a motor, a robot runs a path, a screen shows a message etc. Then, the server sends back to the client a short status response as confirmation. At higher level, coordinated processes should have a manager agent, able to listen low level APIs, evaluate actions and produce orders through API calls to other agents. Thus, the orchestrator can evaluate actions by predefined sequences or reasoning methods as ontologies (OKD-MES) as presented by W. Mohammed on [37]) or prediction methods from artificial intelligence.

This work develops a stand-alone script as the orchestrator agent, to be merged with the general OKD-MES system in future works, just by following the specifications described in this section. The orchestrator contains two TCP/IP interfaces, one to listen HTTP requests from most agents on the environment and other to establish socket connections, for now, exclusively used for YuMi. The programming language selected by the author for the implementation is Python 3, since there are well-documented libraries for RESTful web services, sockets and multiprocessing. Additionally, the SOA allows different programming languages to interact by abstracting their functions into APIs, reachable through a common application interface, in this case a TCP/IP network.

The orchestrator script developed is attached to this document on Appendix A.3. The library used for handling HTTP requests is Flask-Socketio [63], while for Sockets it was used the native Python library also called Sockets [50]. To run both servers in a single script and simplify the agent execution, it was required to use the threading feature of Python, then it was possible to run in parallel a socket listen command to accept Socket connections at port 32200, but also enable an HTTP server listening in port 32100. This technique allows the orchestrator to interact with the ABB robot through sockets, while in parallel, any other HTTP request can be received simultaneously.

The action codes created to communicate with YuMi are described in Table 4.3, these codes are handled by the Socket server aforementioned and designed to interact with the orchestrator component by triggering event notifications or requesting / reading changes on the HMI. These interactions are controlled by variables defined at script scope, above threads. The scope for those variables makes them accessible by both threads. Even

Table 4.3. Action codes for YuMi

| Code | Process | Stage | Remarks |
|---------------------------|---|---|---|
| 0 | Idle | Waiting for start command | Keep socket alive while idle |
| 1 2 3 4 100 | Case assembly | Fastening pieces 1 - 2 Fastening pieces 1 to 3 Fastening pieces 1 to 4 Fastening pieces 1 to 5 Picking pieces | Keep socket alive until done |
| 5 10 11 12 30 | Classifying and packing | Visual inspection Phone model I Phone model H Phone model F Sealing package | Notifications |
| 6 200 300 | Collecting materials Process completed Internal | Picking Products ready to dispatch N/A | Notification Notification Report order size |

threading is a parallel computing paradigm, in this particular case was not necessary to lock the variables, since the system operation does not demand high concurrent access to the common variables. However it must be analyzed in the future if new applications might require locking techniques for the common variables to ensure stability on high speed processes.

The web API for the collaborative workstation contains services and events, in this case all should be invoked by HTTP POST requests. The methodology with services is call on demand, while events work in a push notification method, accessible only for subscribers. Table 4.4 presents the description of events in the collaborative workstation to develop on the web API.

Table 4.4. Description of events for Web APIs linked to the collaborative workstation

| Type | Event ID | Remarks |
|----------|--------------|--|
| Services | PickBasket | Bring the product basket to the collaborative work space. Only Available when YuMi is on "Idle" state. |
| | PickFaces | Bring the case faces to the input conveyor. Only Available when YuMi is on "Idle" state |
| | EnableCollab | Enables the collaborative assembly process. |
| Events | Basketloaded | Basket loaded in the work space |
| | FacesLoaded | Face plans loaded in the work space |
| | Dispatched | Case assembled. |

The HTTP server runs in the main thread of the backend. As explained before, YuMi will

Table 4.5. Web API for services provided by the collaborative workstation

| Service ID | Service URL | Service body |
|--------------|-----------------------------------|--|
| PickBasket | http://(IP)/services/pickbasket | {"orderId":"(order number)", "units":(units)} |
| PickFaces | http://(IP)/services/pickfaces | { } |
| EnableCollab | http://(IP)/services/enablecollab | {"order":["(I),(H),(F)]} |

Table 4.6. Web API to subscribe for events reported by the collaborative workstation

| Event ID | Subscription URL | Subscription body |
|--------------|--|----------------------|
| BasketLoaded | http://(IP)/events/basketloaded/notifs | {"destUrl": "(URL)"} |
| FacesLoaded | http://(IP)/events/facesloaded/notifs | |
| Dispatched | http://(IP)/events/collabdone/notifs | |

interact with this API by common variables accessible by the socket server (secondary thread). The service endpoints for each API accepted by the HTTP server, were implemented using the library Flask. Table 4.5 presents the web API developed. Items on parenthesis (*IP*) on the service URL should be replaced by the IP address of the back-end host. For the service body description, the order number of units carried by the basket and the should be given by the main orchestrator while invoking to pick the basket service. In the case of invoking the service to enable the collaborative workstation, the specific amount of each phone model should be described on H, I and F, regarding section 4.3.

Events are managed on the backend by listening for subscriptions on the main thread. The subscribers list is stored in a local database managed through the SQLAlchemy library [52]. The subscriber list table stores a subscription id, the URL and the event of interest. Then, a method to trigger notifications propagates the message by checking on the database for subscribers of the event triggered. An HTTP POST requests is propagated using the requests library on Python [42], set to do only 1 retry and a timeout of 0.2s, this makes the broadcasting method efficient to be defined as an instant service. All the subscribers on the database receive the instant notification in the following json format:

```
{"class": "eventNotification", "eventID": collabdone, "Info": "(Info)"}
```

The Info parameter contains the order ID for the Basket Loaded and dispatched events, while is empty for the Faces Loaded event. Table 4.6 presents the web API to subscribe for events, the subscription URL has the IP component on parenthesis, this should be replaced by the IP address of the backend host. Besides, the subscription body includes an URL component in parenthesis, this field should be replaced by the URL of the agent subscribed to listen the push notifications generated for the event of interest.

In conclusion, the backend presented can operate as complementary script to the OKD-MES existing in FASTory, since all web APIs enable the human-robot collaborative work-

Table 4.7. List of dependencies required on the virtual environment for the backend

| Package | Version |
|------------------|----------|
| Click | 7 |
| Flask | 1.0.2 |
| Flask-Cors | 3.0.7 |
| Flask-SQLAlchemy | 2.4.1 |
| Flask-SocketIO | 3.3.2 |
| Jinja2 | 2.1 |
| MarkupSafe | 1.1.1 |
| SQLAlchemy | 1.3.8 |
| Werkzeug | 0.15.1 |
| asn1crypto | 1.0.0 |
| certifi | 2019.3.9 |
| cffi | 1.12.3 |
| chardet | 3.0.4 |
| cryptography | 2.7 |
| gevent | 1.4.0 |
| greenlet | 0.4.15 |
| http-ece | 1.1.0 |
| idna | 2.8 |
| itsdangerous | 1.1.0 |
| pip | 9.0.1 |
| py-vapid | 1.7.0 |
| pycparser | 2.19 |
| python-engineio | 3.5.0 |
| python-socketio | 3.1.2 |
| pywebpush | 1.10.0 |
| requests | 2.21.0 |
| setuptools | 28.8.0 |
| six | 1.12.0 |
| urllib3 | 1.24.2 |

station to be accessible through RESTful requests. This component can be executed on the orchestrator server or any other physical device able to execute Python 3 and connect to the Ethernet network in FAST-Lab. The script developed and presented on Annex A.3 was implemented on debugger environment of Flask. Later, the backend will require specific actions to deploy it for production environment. Finally the dependencies for the virtual environment are presented on Table 4.7.

To conclude, the specific execution order recommended to startup the collaborative workstation is:

1. Run the backend (hosted by default on 192.168.125.208:32100)
2. Open the HMI by accessing the backend host address
3. Run the program on the production window at YuMi

4.5 Human Machine Interface

The interactive device available in FAST-Lab to connect the human operator and the cobot is a 12 inch tablet, with Windows 10 OS pre-installed. The tablet has a dedicated slot in the work space, located in an easy-accessible area of the collaborative workspace (low-right corner). There, the operator will be able to see information and interact picking commands by virtual buttons in the touchscreen.

The HMI can be accessed as a website, developed on HTML, using the CSS example "Screen 50/50 Template" from W3 schools [58]. The web application model followed is client-server, using socket.io as interface to enable real-time, bidirectional and event-based communication. The backend serves the website using the main thread and the client can be a web browser pointing to the host IP address (use the IP as URL). The tablet must be connected to the FAST-Lab network and to internet to render properly the HMI. The HMI was tested successfully on the tablet using Google Chrome version 77.0.3865.90 (64-bit) and Mozilla Firefox 69.0.2 (64-bit).

The HMI design is presented on Figure 4.11. On the left white section, there are useful links to access the FAST-Lab website and a lower section with temporary buttons to simulate requests from the OKD-MES. The purple section in the middle reports the process ongoing and the current task in execution. It also contains the button to start the collaborative assembly process. The right section contains a demonstrative video about the assembly process, provided by L. Amezuza from [3]. Below the video, a transition button to advance stage appears while the human operator and cobot are assembling faces, it allows the operator to confirm the current task was completed.

The code presented on Appendix A.4 includes dependencies and a script section (lines 10 to 61), where the Socket.io component is implemented using Javascript programming language. The script has actions on click for every button on the interface. It is linked to a socket emit event and the general namespace used is */HMI*. Then some Socket.io receiver components describe the actions taken on the HMI every time the backend call them. This structure is mirrored on the backend, where the events can be invoked by the counterpart. The list of events for each listener side is presented on Table 4.8.

4.6 Adjacent devices

As explained on Chapter 3, the new FAST-Lab extends its scope by creating new research environments. The new environments contain equipment required to interoperate

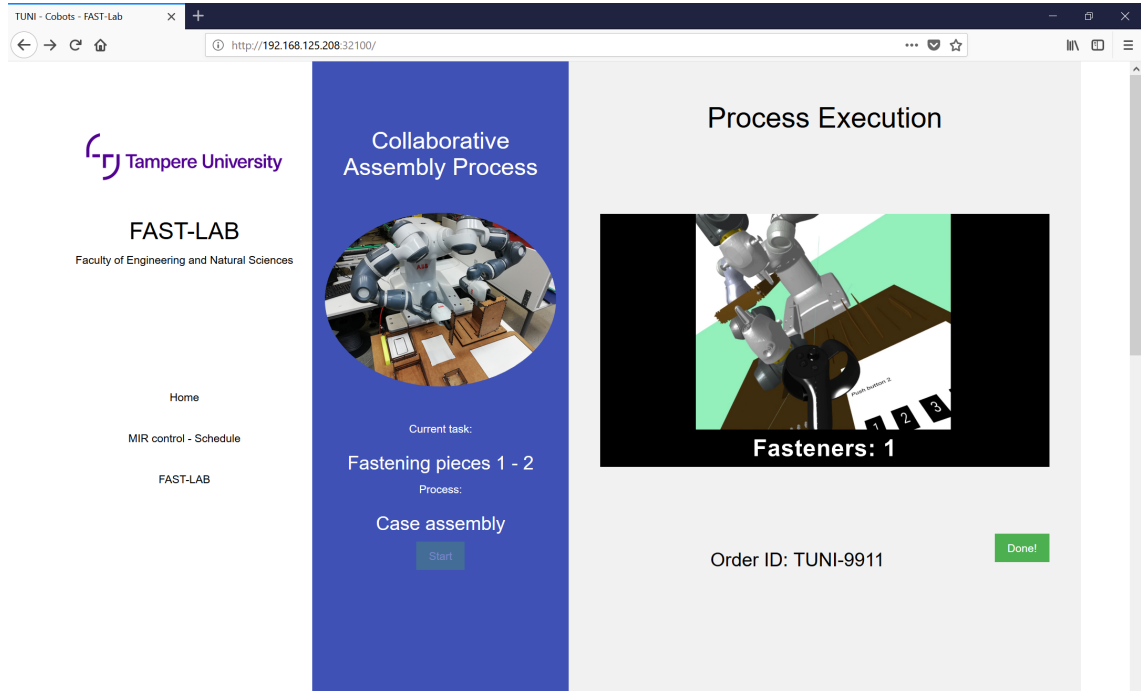


Figure 4.11. Capture of the visual design for the Human Machine Interface in the collaborative workstation

by merging the existing orchestration system. The human-robot collaborative workstation requires to get supplies carried by the AMIR. Besides, the case parts must be stamped and palletized by a dual robot cell (ABB IRB 140), located behind the collaborative workstation. To demonstrate the interoperation capabilities on this environment, is not enough to develop web APIs, but also is required to setup the adjacent devices to match the process requirements described on this work.

The conveyors located on each side of the collaborative workstation, transport the case parts between the human-robot collaborative workstation and the dual robot cell. The left conveyor, also called *plans conveyor*, receives the case plans brought by AMIR and discharged by YuMi, while the right conveyor, also called *pallet conveyor*, transports a pallet with all the case faces organized by the dual robot cell, for further collaborative assembly. Both conveyors are equipped by INICO S1000 and the motion drive is an OMRON Sysdrive 3G3JV. The INICO acts as RTU, providing digital commands to the motion drive. Each drive can start, stop and select the displacement direction of the respective conveyor.

The web APIs developed for the conveyors are described on Table 4.9 and specified on Table 4.10. On the API specification table, the IP term in parenthesis should be replaced for the INICO S1000 IP address, by default, the addresses are 192.168.2.54 and 192.168.3.57 for the faces and pallet conveyor respectively. To subscribe for the events the URL term should be replaced for the URL of the agent subscribed to listen the push notifications from the conveyor. The Json format of the event messages produced by the conveyors is:

Table 4.8. *Interactive events for the HMI*

| Listener side | Event ID | Description |
|---------------|------------|---|
| Backend | next | Event controlling the process flow trigger for the collaborative assembly process |
| | connect | Event triggered every time the client access the website. This starts the secondary thread, enabling the backend to connect with YuMi. Without any HMI client connected is impossible to establish any control link with YuMi |
| | disconnect | Spare event for actions when the website is closed |
| | supply | Provisional event to call the MIR robot to the collaborative workstation |
| | pickup | Provision event to send away the MIR robot to the loading point at FASTory line |
| HMI | status | Event containing the informative text to be displayed on the task and process area |
| | idle | Event confirming the robot is on-line. Enables the Start button |
| | human | Event to disable any transition button |
| | finish | Event to hide the transition button and enable the start button |
| | video | Event to update the demonstrative video depending the assembly stage controlled by the backend |

{"senderID": "(Conveyor name)","Status":"(Loaded or Unloaded)"}

The functionality programmed for the faces conveyor is only to load the face plans on the dual robot cell by service request. After stamping and palletizing, the cell can drop away the plans without faces. For the pallet conveyor the functionality is based on using a unique pallet to transport the faces. The pallet can move towards the collaborative workstation by a service request and can be reloaded automatically to the dual robot cell, by placing it on the loading sensor, located and marked on the conveyor. The cell will report when is loaded and unloaded. The human operator on the collaborative workstation is in charge of removing the pallet from the conveyor and place it correctly on the slot at the collaborative work space. For better detail please refer to Annex A.5 and A.6, which presents the structured text code implemented on each INICO S1000 RTU.

Regarding the AMIR, all the interaction is based on web APIs managed by the orchestrator, but exceptionally and for demonstrative purposes, the HMI developed on this work included provisional buttons to call the AMIR to the collaborative workstation and simulate the FASTory operation. Two missions were programmed on the MIR-100 Robot. One mission order the robot to navigate to the collaborative workstation, while the second mission sends the robot away to FASTory. The MIR-100 provides a built in web platform

Table 4.9. Description of services and events for Web APIs linked to the side conveyors

| Type | ID | Remarks |
|----------|-----------|--|
| Services | Load | Only for plans conveyor. Drives the faces from the collaborative workstation to the dual robot cell. The presence sensor on the collaborative workstation side must be activated and the sensor on the dual robot cell must be deactivated. Otherwise, the order will not be executed |
| | Unload | Only for pallet conveyor. Drives the faces palletized from the dual robot cell to the collaborative workstation. The presence sensor on the dual robot cell must be activated and the sensor on the collaborative workstation must be deactivated. Otherwise, the order will not be executed |
| Events | CNVStatus | Triggered every time the conveyor completes a load or unload action. Contains status value as a "Status" attribute on Json |

Table 4.10. Web API for services and events provided by the conveyors

| Type | ID | URL | Body |
|---------|-----------|-------------------------------------|----------------------|
| Service | Load | http://(IP)/services/load | { } |
| | Unload | http://(IP)/services/unload | |
| Event | CNVStatus | http://(IP)/events/CNVStatus/notifs | {"destUrl": "(URL)"} |

where all the missions can be created and web APIs can be associated. In that sense, no development is required for this task, different than use the web interface to program the missions and acquire the authorization keys and URLs to invoke the missions. Because of several changes carried on FAST-Lab during the elaboration of this work, the author considered irrelevant to include in this document any visual reference of the missions created on the MIR-100 platform, because the content might turn obsolete in short time.

4.7 Testing method

To test the operation of the collaborative workstation implemented in FAST-Lab, 2 volunteers will act as human operators and will try the new workstation by executing the actions highlighted in the flowchart on Figure 4.12 [6], which describes roughly the extended process on FASTory. The actions highlighted integrate a testing process, planned to take place on the real environment at FAST-Lab. Each person will repeat the assembly task 25 times, under observation and tutoring by the author. The test will measure accuracy by finding the percentage of executions which concluded the whole testing process successfully. Then, the precision will be assessed by finding the percentage of executions

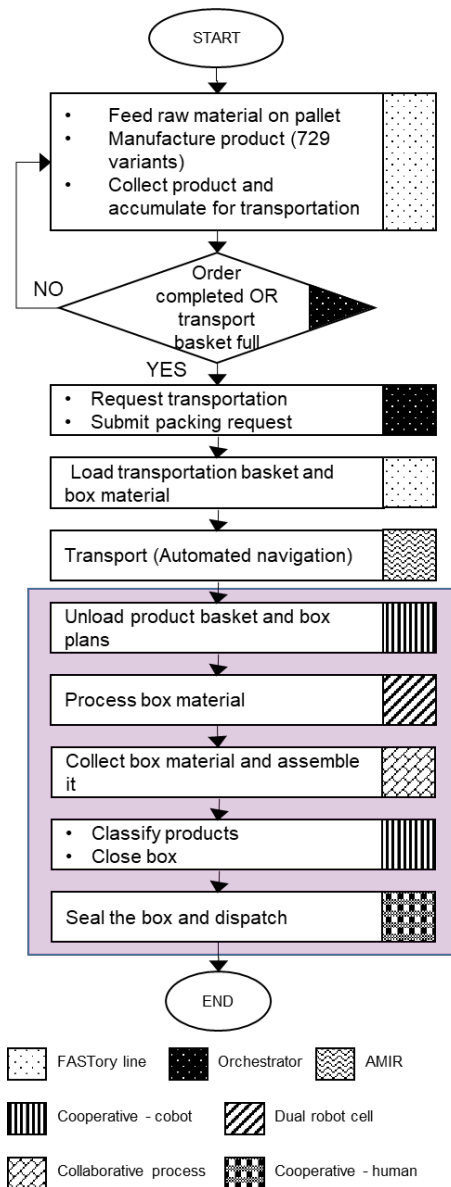


Figure 4.12. Flowchart for the testing process

where the operator need does not need to adapt manually the pieces to match the tabs. Finally the time efficiency will be assessed by calculating the percentage difference between the time spent to assembly the modular case for a person without a cobot and the time spent on the human-robot collaborative workstation to complete the same process.

By evaluating accuracy, precision and time efficiency will be possible analyse and conclude the benefits and challenges of implementing a human-robot collaborative assembly workstation.



Figure 4.13. *Final implementation of the human-robot collaborative workstation at FAST-Lab*

4.8 Results

The final register of the workstation implemented on FAST-Lab is presented on Figure 4.13. Overall, the results presented an average accuracy of 95 % and average precision of 72 %. Values are considered acceptable by the author, since the testers expressed satisfaction and comfort after completing the total amount of testing iterations. However, the author detected that human intervention has considerable impact on the handling precision of the workpiece, since involuntarily, the operator generated drifting on the faces grasped by the robot.

The average execution time for one of the operators alone is 255 secs, in contrast with 358 secs, spent by the cobot collaborating to the same operator, this represents an increase of roughly 40% the process time. A comparative compendium of results is presented on Figure 4.14, where is possible to identify the time efficiency as the best opportunity to improve.

Apart from numerical results, the perception from the user and even developer had a notorious impact on the way how this implementation was completed. Methodologies as EXKB and RMAT demonstrate that kinematic solutions by automated mathematical processing are not enough to satisfy the expectations of users and owners of cobots in a coexistent and simultaneous work situation. This provide a very important feedback for future research, since regular robots follow formally an optimization paradigm to reduce

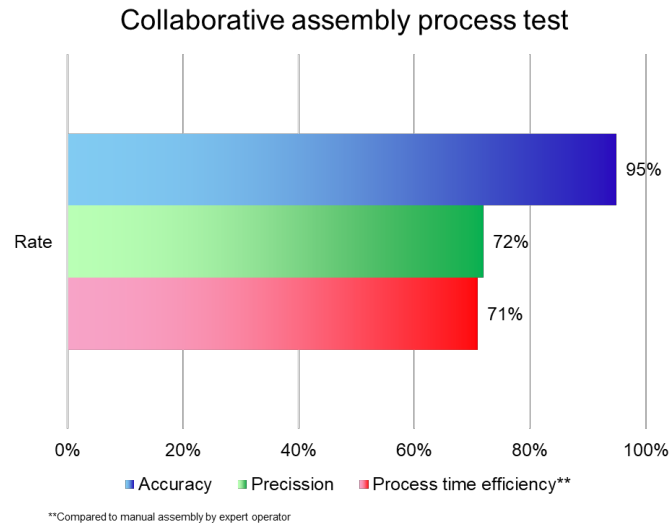


Figure 4.14. Comparative bar graph of results in accuracy, precision and process time

| | | | |
|-----------|--|---|-----------|
| Advantage | Cobots simplify the role of humans in manufacturing | Cobots require high economic investment, compared with human labor costs | Challenge |
| | Safe simultaneous coexistence of robot and human | Cobots have reduced agility and force capacity | |
| | Guidelines for better human – robot coexistence are mature | New technologies should focus on interaction with human behaviors (operator) | |
| | Cobots reduce the technological gap between conventional robot skills and human dexterity | Humans will need to focus in the future on knowledge with more impact and enhance their human-exclusive skills | |
| | Cobots facilitate programming by demonstration. I.e. lead-through or pose sculpt | Motion controllers in cobots are required to solve and execute more difficult paths including dynamic context aware and human behavior. This might require use of AI in the future. | |
| | Feasibility of cooperative and collaborative work between humans and robots is demonstrated | Additional development is needed to match capabilities of conventional robots, as agility and force, without compromising human safety | |
| | Cobots can include any additional feature beside manufacturing aid, to improve the user experience | Commercial models require to include better interfaces to achieve empathy / comfort / esthetics that satisfy and engage human users | |
| | Accuracy level is acceptable for basic collaborative assembly tasks | Zero Defect Manufacturing strategies should focus on precision and process time of cobots | |

Figure 4.15. Advantages and challenges of implementing cobots by exemplifying a real industrial scenario of human-robot collaboration for assembly

process time and demand less operative resources than humans. But when is about collaborative work, the comfort, confidence and empathy felt by the human might play important role that influence changes on technological tools. Figure 4.15 summarizes the main aspects from all the qualitative observations collected during the testing stage [6].

This section presented a compilation of observations and experiments carried in FAST-

Lab to conceive all the technical specifications (detailed engineering) providing effective operational methods for the new collaborative workstation and its new interconnected environment.

5 CONCLUSIONS

This document presented the design and reported the implementation of a human-robot collaborative assembly workstation, integrated to the FASTory modular robotized production line at FAST-Lab.

The activities carried, allowed the author to select and design a demonstrative human-robot collaborative assembly process, based in a modular tabbed case as work piece. In this way is possible to demonstrate Human - Robot Collaborative Manufacturing. This activity extended the FASTory process, providing a packing stage for the printouts already produced by the line. Lately, it was possible to provide technical details about how to design and implement a human-robot collaborative assembly workstation, aiming to reproduce the assembly process of a modular case.

The prototype built over detailed engineering specifications, presented how to integrate the human-robot collaborative workstation to a modular interconnected production line, using SOA by means of a software component developed in Python 3 programming language. The script (backend) used tools from different software engineering topics such as multi threading, web applications, databases, multi agent systems etc.

Finally, observations of the implementation and results of an operative testing, provided information about how accurate and precise is the operation of the cobot. But also presented drawbacks as reduction on processing time. The compilation of empirical observations was used to assess the advantages and challenges of implementing a human-robot collaborative assembly workstation, providing an overview of the advantages and challenges of current HRC technologies at the shop floor

Besides reporting the specifications and the test of the human-robot collaborative assembly workstation implemented in FAST-Lab, this work aims to provide a better definition of "collaborative work" between a human operator and a dual arm cobot. Many existing works present "collaboration" as a process composed by sequential action (shifted), without fully simultaneous and coexistent tasks performed by robots and humans, while research focus make emphasis on the benefits of join humans and robots equally for industrial applications.

In general, literature about cobots present collaborative work as the summation of actions of humans and robots, which correspond to a former definition of cobots, dedicated to serve on mechanical tasks guided by humans using direct control. However, the collaborative process presented in this work, demonstrates that the imaginary of cobots have

multiple dimensions. For standardization, the interests points to prescribe features of a robot that fulfill safe operation for human interaction, breaking the traditional paradigm of reduce risk by limiting exposure (fences and indication). While for research dimension, the interest aims to explore deeper the opportunities of merging humans and robots in future manufacturing.

This implementation demonstrated the positive feasibility to implement cobots in future manufacturing scenarios, integrated to a modular production line enabled by multiple ICT technologies. The benefit of including machines managing information at the plant floor reduce possible human errors and creates new perspectives on the changes predicted as impact of Industry 4.0. During the the experimental phase, the author questioned about the idea of replacing RTUs for middleware or software addapters to facilitate the integration of legacy equipment into SOA systems. This situation might predict a future scenario of ICT technologies increasing its penetration into the industrial automation sector. In the near future, even obsolete devices will provide Ethernet support, making them cost effective and easier to integrate by software means.

From another perspective, the figure of orchestrators and SOA should be prepared to host new information protocols and communication methods, independently of hardware brands, programming languages or network technologies.

5.1 Supported and future works

By the date of submission, this work supported another research works providing a framework for the Master's thesis presented on [56] and [3] by A. Toichoa and L. Amezua respectively. Also another research assignment titled "An approach for adapting a cobot workstation to human operator within a deep learning camera" presented by O. de Miguel at FAST-Lab, included the human-robot collaborative workstation as research tool. The works mentioned adopted the collaborative process presented on this document and enriched the implementation adding features as a prototype for emotion-driven interaction, adaptive process parameters based on visual recognition for the operator and virtual reality environments for training. These supported works provided the head model installed on YuMi (4.13) and the demonstrative videos included on the HMI (Figure 4.11).

The following topics are suggested as future works:

- Implement of a context-aware error handler for dual arm cobots. This might contribute to enhance the accuracy of workstations using cobots, by providing actions on exceptional situations
- Update for the OKD-MES system in FASTory, to included the new APIs developed in this work. This might allow to merge the collaborative workstation, the AMIR, FASTory, the material handling cell and any other future component in FAST-Lab
- Enhance grasping methods for cobots. As concluded, the influence of human ma-

nipulation in robotized process has a negative impact on precision by involuntary drifting of work pieces. This can be addressed by proposing enhanced grasping methods specific for cobots

- Implement a stamping and palletizing process for the modular case parts at the dual robot cell. The dual robot cell integrated to the human-robot collaborative workstation will require to develop the integration methods similar as the ones used in the backend, but for conventional robots
- Implement a solution for classifying and storing products discarded to the buffer basket on the human-robot collaborative workstation. The products collected in the buffer basket will require future classification and storage
- Assess the application of cyber-security methodologies for the human-robot collaborative workstation. The implementation did not focused on cyber-security for industrial technological assest. This can improve the realism on the demonstrative environment
- Assess the ECKB and RMat methodologies in new contexts
- Assess the impact of human-associated variables in cobots programming. The relevance of trust, comfort and aesthetics on human-robot collaborative workstations play an important role on technical aspects. The proper assessment of subjective variables attached to humans, might drive to specify successfully technical details of workstations including cobots.

This document was elaborated in \LaTeX using the template provided by Tampere University on its version 1.3 by Ville Koljonen. The content was edited using the Overleaf environment.

REFERENCES

- [1] *ABB's Collaborative Robot -YuMi - Industrial Robots From ABB Robotics*. en. URL: <https://new.abb.com/products/robotics/industrial-robots/irb-14000-yumi> (visited on 09/15/2019).
- [2] S. O. Afolaranmi, B. R. Ferrer and J. L. M. Lastra. A Framework for Evaluating Security in Multi-Cloud Environments. *IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society*. Oct. 2018, 3059–3066. DOI: 10.1109/IECON.2018.8591454.
- [3] L. Amezua Hormaza. *A virtual reality environment for training operators for assembly tasks involving human-cobot interactions*. eng. 2019.
- [4] *ANSI/RIA R15.06-2012 - Industrial Robots and Robot Systems - Safety Requirements*. URL: <https://webstore.ansi.org/Standards/RIA/ANSIRIAR15062012> (visited on 02/13/2019).
- [5] C. Bartneck and J. Forlizzi. A design-centred framework for social human-robot interaction. *RO-MAN 2004. 13th IEEE International Workshop on Robot and Human Interactive Communication (IEEE Catalog No.04TH8759)*. Sept. 2004, 591–594. DOI: 10.1109/ROMAN.2004.1374827.
- [6] R. Bejarano. Implementing a Human-Robot Collaborative Assembly Workstation (Accepted). Espoo, Finland, July 2019.
- [7] D. Bortot, M. Born and K. Bengler. Directly or on detours? How should industrial robots approximate humans?: *ACM/IEEE International Conference on Human-Robot Interaction*. 2013. ISBN: 978-1-4673-3055-8. DOI: 10.1109/HRI.2013.6483515.
- [8] C. Breazeal. Designing Sociable Robots. *Designing Sociable Robots*. MITP, 2004. ISBN: 978-0-262-25583-7. URL: <https://ieeexplore.ieee.org/document/6280040> (visited on 01/31/2019).
- [9] BSI. *ISO 10218-1:2011- Robots and robotic devices — Safety requirements for industrial robots, Part 1: Robots*. 2014. DOI: 10.1016/j.joi.2008.08.001.
- [10] D. Busson, R. Bearee and A. Olabi. Task-oriented rigidity optimization for 7 DOF redundant manipulators. en. *IFAC-PapersOnLine* 50.1 (July 2017), 14588–14593. ISSN: 24058963. DOI: 10.1016/j.ifacol.2017.08.2108. URL: <https://linkinghub.elsevier.com/retrieve/pii/S2405896317327726> (visited on 09/09/2019).
- [11] D. Busson and R. Bearee. A Pragmatic Approach to Exploiting Full Force Capacity for Serial Redundant Manipulators. *IEEE Robotics and Automation Letters* 3.2 (Apr. 2018), 888–894. ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2018.2792541. URL: <http://ieeexplore.ieee.org/document/8255624/> (visited on 09/09/2019).
- [12] S. M. Casner, E. L. Hutchins and D. Norman. The Challenges of Partially Automated Driving. *Commun. ACM* 59.5 (Apr. 2016), 70–77. ISSN: 0001-0782. DOI:

- 10.1145/2830565. URL: <http://doi.acm.org/10.1145/2830565> (visited on 01/31/2019).
- [13] A. Cherubini, R. Passama, A. Crosnier, A. Lasnier and P. Fraisse. Collaborative manufacturing with physical human–robot interaction. *Robotics and Computer-Integrated Manufacturing* 40 (Aug. 2016), 1–13. ISSN: 0736-5845. DOI: 10.1016/j.rcim.2015.12.007. URL: <http://www.sciencedirect.com/science/article/pii/S0736584515301769> (visited on 02/03/2019).
 - [14] *Collaborative Robot Market| Cobot Market Study | Cobot Intelligence*. en-US. URL: <https://cobotintel.com/guide-to-collaborative-robots-market> (visited on 09/07/2019).
 - [15] *ColRobot Demonstration in the Aerospace Industry*. Jan. 2018. URL: https://www.youtube.com/watch?time_continue=167&v=BbKqAB_gLfg (visited on 09/09/2019).
 - [16] A. M. Dabrowski. *Geometric Abstraction | Essay | Heilbrunn Timeline of Art History | The Metropolitan Museum of Art*. en. URL: https://www.metmuseum.org/toah/hd/geab/hd_geab.htm (visited on 09/12/2019).
 - [17] D. G. U. e.V. *IFA - Technical information: Collaborative robots (COBOTS)*. EN. URL: <https://www.dguv.de/ifa/fachinfos/kollaborierende-roboter/index-2.jsp> (visited on 02/01/2019).
 - [18] *FASTory Simulator*. URL: <http://escop.rd.tut.fi:3000/> (visited on 09/11/2019).
 - [19] B. R. Ferrer, W. M. Mohammed, A. Lobov, A. M. Galera and J. L. M. Lastra. Including human tasks as semantic resources in manufacturing ontology models. *IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society*. Oct. 2017, 3466–3473. DOI: 10.1109/IECON.2017.8216587.
 - [20] *Flexible programming and orchestration of collaborative robotic manufacturing systems - IEEE Conference Publication*. URL: <https://ieeexplore.ieee.org/abstract/document/8472058> (visited on 02/08/2019).
 - [21] S. A. Green, M. Billingham, X. Chen and J. G. Chase. Human-Robot Collaboration: A Literature Review and Augmented Reality Approach in Design. en. *International Journal of Advanced Robotic Systems* 5.1 (Mar. 2008), 1. ISSN: 1729-8814. DOI: 10.5772/5664. URL: <https://doi.org/10.5772/5664> (visited on 02/08/2019).
 - [22] J. Guérin, O. Gibaru, S. Thiery and E. Nyiri. Locally optimal control under unknown dynamics with learnt cost function: application to industrial robot positioning. *Journal of Physics: Conference Series* 783 (Jan. 2017), 012036. ISSN: 1742-6588, 1742-6596. DOI: 10.1088/1742-6596/783/1/012036. URL: <http://stacks.iop.org/1742-6596/783/i=1/a=012036?key=crossref.361aa0aebc80777430ff78a1637784c8> (visited on 09/09/2019).
 - [23] J. GuErin, S. Thiery, E. Nyiri and O. Gibaru. Unsupervised Robotic Sorting : Towards Autonomous Decision Making Robots. *International Journal of Artificial Intelligence & Applications* 9.2 (Mar. 2018), 81–98. ISSN: 09762191, 0975900X. DOI: 10.5121/ijaia.2018.9207. URL: <http://aircconline.com/ijaia/V9N2/9218ijaia07.pdf> (visited on 09/09/2019).

- [24] K. B. Head. *Robots and humans can work together with new ISO guidance*. en. URL: <http://www.iso.org/cms/render/live/en/sites/isoorg/contents/news/2016/03/Ref2057.html> (visited on 02/03/2019).
- [25] V. Herrera, A. Bepperling, A. Lobov, H. Smit, A. W. Colombo and J. L. Lastra. Integration of Multi-Agent Systems and Service-Oriented Architecture for industrial automation. *IEEE International Conference on Industrial Informatics (INDIN)*. 2008. ISBN: 978-1-4244-2171-8. DOI: 10.1109/INDIN.2008.4618205.
- [26] S. Iarovyi, X. Xu, A. Lobov, J. L. Martinez Lastra and S. Strzelczak. Architecture for Open, Knowledge-Driven Manufacturing Execution System. en. *Advances in Production Management Systems: Innovative Production Management Towards Sustainable Growth*. Ed. by S. Umeda, M. Nakano, H. Mizuyama, H. Hibino, D. Kiritsis and G. von Cieminski. IFIP Advances in Information and Communication Technology. Springer International Publishing, 2015, 519–527. ISBN: 978-3-319-22759-7.
- [27] *IEC 61508-1:2010 | IEC Webstore | functional safety, smart city*. URL: <https://webstore.iec.ch/publication/5515> (visited on 02/03/2019).
- [28] ISO 10218-2. (ISO 10218-2:2011) Robots and robotic devices – Safety requirements for industrial robots – Part 2: Robot systems and integration. *Text* (2011). ISSN: 2831889189. DOI: 10.5594/J09750.
- [29] Z. Liandong and W. Qifeng. Integrated Collaborative Manufacturing Management System for Complex Product. *2009 Second International Conference on Future Information Technology and Management Engineering (FITME 2009)(FITME)*. Dec. 2010, 206–209. ISBN: 978-1-4244-5339-9. DOI: 10.1109/FITME.2009.57. URL: doi.ieeecomputersociety.org/10.1109/FITME.2009.57 (visited on 02/08/2019).
- [30] P. Lin, K. Abney and G. Bekey. Robot ethics: Mapping the issues for a mechanized world. *Artificial Intelligence*. Special Review Issue 175.5 (Apr. 2011), 942–949. ISSN: 0004-3702. DOI: 10.1016/j.artint.2010.11.026. URL: <http://www.sciencedirect.com/science/article/pii/S0004370211000178> (visited on 01/31/2019).
- [31] I. Lopez-Juarez, J. Corona-Castuera, M. Peña-Cabrera and K. Ordaz-Hernandez. On the design of intelligent robotic agents for assembly. *Information Sciences*. Intelligent Embedded Agents 171.4 (May 2005), 377–402. ISSN: 0020-0255. DOI: 10.1016/j.ins.2004.09.011. URL: <http://www.sciencedirect.com/science/article/pii/S0020025504003159> (visited on 02/02/2019).
- [32] *MakerCase - Easy Laser Cut Case Design*. URL: <http://old.makercase.com/> (visited on 09/12/2019).
- [33] I. E. Makrini, K. Merckaert, D. Lefebvre and B. Vanderborght. Design of a collaborative architecture for human-robot assembly tasks. *IEEE International Conference on Intelligent Robots and Systems*. 2017. ISBN: 978-1-5386-2682-5. DOI: 10.1109/IRoS.2017.8205971.
- [34] N. Mendes, J. Ferrer, J. Vitorino, M. Safeea and P. Neto. Human Behavior and Hand Gesture Classification for Smart Human-robot Interaction. en. *Procedia Manufacturing* 11 (2017), 91–98. ISSN: 23519789. DOI: 10.1016/j.promfg.2017.07.156.

- URL: <https://linkinghub.elsevier.com/retrieve/pii/S2351978917303621> (visited on 09/09/2019).
- [35] G. Michalos, S. Makris, P. Tsarouchi, T. Guasch, D. Kontovrakis and G. Chrysosouris. Design considerations for safe human-robot collaborative workplaces. *Procedia CIRP*. 2015. ISBN: 978-1-5108-1521-6. DOI: 10.1016/j.procir.2015.08.014.
 - [36] W. M. Mohammed, A. Lobov, B. R. Ferrer, S. Iarovyi and J. L. M. Lastra. A web-based simulator for a discrete manufacturing system. *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*. Oct. 2016, 6583–6589. DOI: 10.1109/IECON.2016.7793563.
 - [37] W. M. Mohammed. *Encapsulation of MES Functionalities as RESTful Web Services for Knowledge-Driven Manufacturing Systems*. eng. 2017.
 - [38] L. Pazzi and M. Pellicciari. From the Internet of Things to Cyber-Physical Systems: The Holonic Perspective. en. *Procedia Manufacturing* 11 (2017), 989–995. ISSN: 23519789. DOI: 10.1016/j.promfg.2017.07.204. URL: <https://linkinghub.elsevier.com/retrieve/pii/S2351978917304122> (visited on 09/09/2019).
 - [39] M. A. Peshkin, J. E. Colgate, W. Wannasuphoprasit, C. A. Moore, R. B. Gillespie and P. Akella. Cobot architecture. *IEEE Transactions on Robotics and Automation* 17.4 (Aug. 2001), 377–390. ISSN: 1042-296X. DOI: 10.1109/70.954751.
 - [40] *Product specification - IRB 14000*. URL: <https://search-ext.abb.com/library/Download.aspx?DocumentID=3HAC052982-001&LanguageCode=en&DocumentPartId=&Action=Launch> (visited on 09/15/2019).
 - [41] B. Ramis, L. Gonzalez, S. Iarovyi, A. Lobov, J. L. Martinez Lastra, V. Vyatkin and W. Dai. Knowledge-based web service integration for industrial automation. *Proceedings - 2014 12th IEEE International Conference on Industrial Informatics, INDIN 2014*. 2014. ISBN: 978-1-4799-4905-2. DOI: 10.1109/INDIN.2014.6945604.
 - [42] *Requests: HTTP for HumansTM — Requests 2.22.0 documentation*. URL: <https://requests.kennethreitz.org/en/master/> (visited on 10/05/2019).
 - [43] L. Rozo, S. Calinon, D. G. Caldwell, P. Jiménez and C. Torras. Learning Physical Collaborative Robot Behaviors From Human Demonstrations. *IEEE Transactions on Robotics* 32.3 (June 2016), 513–527. ISSN: 1552-3098. DOI: 10.1109/TR0.2016.2540623.
 - [44] J. Saenz, C. Vogel, F. Penzlin and N. Elkmann. Safeguarding Collaborative Mobile Manipulators - Evaluation of the VALERI Workspace Monitoring System. en. *Procedia Manufacturing* 11 (2017), 47–54. ISSN: 23519789. DOI: 10.1016/j.promfg.2017.07.129. URL: <https://linkinghub.elsevier.com/retrieve/pii/S2351978917303335> (visited on 09/09/2019).
 - [45] M. Safeea, N. Mendes and P. Neto. Minimum Distance Calculation for Safe Human Robot Interaction. en. *Procedia Manufacturing* 11 (2017), 99–106. ISSN: 23519789. DOI: 10.1016/j.promfg.2017.07.157. URL: <https://linkinghub.elsevier.com/retrieve/pii/S2351978917303633> (visited on 09/09/2019).

- [46] M. Sawhney. *As Robots Threaten More Jobs, Human Skills Will Save Us*. en. URL: <https://www.forbes.com/sites/mohanbirsawhney/2018/03/10/as-robots-threaten-more-jobs-human-skills-will-save-us/> (visited on 01/31/2019).
- [47] B. Siciliano and O. Khatib, eds. *Springer Handbook of Robotics*. en. Berlin Heidelberg: Springer-Verlag, 2008. ISBN: 978-3-540-30301-5. URL: <http://www.springer.com/gp/book/9783540303015> (visited on 01/29/2019).
- [48] C. Smith, Y. Karayiannidis, L. Nalpantidis, X. Gratal, P. Qi, D. V. Dimarogonas and D. Kragic. Dual arm manipulation—A survey. *Robotics and Autonomous Systems* 60.10 (Oct. 2012), 1340–1353. ISSN: 0921-8890. DOI: 10.1016/j.robot.2012.07.005. URL: <http://www.sciencedirect.com/science/article/pii/S092188901200108X> (visited on 02/20/2019).
- [49] H. Sobreira, C. M. Costa, I. Sousa, L. Rocha, J. Lima, P. C. M. A. Farias, P. Costa and A. P. Moreira. Map-Matching Algorithms for Robot Self-Localization: A Comparison Between Perfect Match, Iterative Closest Point and Normal Distributions Transform. en. *Journal of Intelligent & Robotic Systems* 93.3-4 (Mar. 2019), 533–546. ISSN: 0921-0296, 1573-0409. DOI: 10.1007/s10846-017-0765-5. URL: <http://link.springer.com/10.1007/s10846-017-0765-5> (visited on 09/09/2019).
- [50] *socket — Low-level networking interface — Python 3.7.5rc1 documentation*. URL: <https://docs.python.org/3/library/socket.html> (visited on 10/02/2019).
- [51] G. B. d. Sousa, A. Olabi, J. Palos and O. Gibaru. 3D Metrology Using a Collaborative Robot with a Laser Triangulation Sensor. en. *Procedia Manufacturing* 11 (2017), 132–140. ISSN: 23519789. DOI: 10.1016/j.promfg.2017.07.211. URL: <https://linkinghub.elsevier.com/retrieve/pii/S2351978917304195> (visited on 09/09/2019).
- [52] *SQLAlchemy - The Database Toolkit for Python*. URL: <https://www.sqlalchemy.org/> (visited on 10/05/2019).
- [53] S. Stadler, A. Weiss, N. Mirnig and M. Tscheligi. Anthropomorphism in the factory - A paradigm change?: *ACM/IEEE International Conference on Human-Robot Interaction*. 2013. ISBN: 978-1-4673-3055-8. DOI: 10.1109/HRI.2013.6483586.
- [54] H. Tang, X. Cao, A. Song, Y. Guo and J. Bao. Human-robot collaborative teleoperation system for semi-autonomous reconnaissance robot. *2009 International Conference on Mechatronics and Automation*. Aug. 2009, 1934–1939. DOI: 10.1109/ICMA.2009.5246589.
- [55] *Technical reference manual IRAPID Instructions, Functions and Data*. URL: <https://search-ext.abb.com/library/Download.aspx?DocumentID=9AKK107046A8697&LanguageCode=en&DocumentPartId=&Action=Launch> (visited on 09/30/2019).
- [56] A. Toichoa Eyam. *Emotion-driven human-cobot interaction based on EEG in industrial applications*. eng. 2019.
- [57] R. del Toro, M. C. Schmittiel, R. E. Haber Guerra and R. Haber-Haber. System Identification of the High Performance Drilling Process for Network-Based Control. Jan. 2008, 827–834. DOI: 10.1115/DETC2007-34307.

- [58] *Tryit Editor v3.6*. en-US. URL: https://www.w3schools.com/w3css/tryit.asp?filename=tryw3css_templates_fifty&stacked=h (visited on 10/05/2019).
- [59] V. Villani, F. Pini, F. Leali and C. Secchi. Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications. *Mechatronics* 55 (Nov. 2018), 248–266. ISSN: 0957-4158. DOI: 10.1016/j.mechatronics.2018.02.009. URL: <http://www.sciencedirect.com/science/article/pii/S0957415818300321> (visited on 02/03/2019).
- [60] H. Wali. *Human Robot Collaboration in Assembly Processes*. eng. 2018.
- [61] W. Wannasuphoprasit, R. B. Gillespie, J. E. Colgate and M. A. Peshkin. Cobot control. *Proceedings of International Conference on Robotics and Automation*. Vol. 4. Apr. 1997, 3571–3576 vol.4. DOI: 10.1109/ROBOT.1997.606888.
- [62] H. Wei, Z. Shao, Z. Huang, R. Chen, Y. Guan, J. Tan and Z. Shao. RT-ROS: A real-time ROS architecture on multi-core processors. *Future Generation Computer Systems* 56 (Mar. 2016), 171–178. ISSN: 0167-739X. DOI: 10.1016/j.future.2015.05.008. URL: <http://www.sciencedirect.com/science/article/pii/S0167739X15001831> (visited on 09/22/2019).
- [63] *Welcome to Flask-SocketIO's documentation! — Flask-SocketIO documentation*. URL: <https://flask-socketio.readthedocs.io/en/latest/> (visited on 10/02/2019).
- [64] *What is a TCP/IP Socket Connection ?* en-US. Oct. 2014. URL: www.ibm.com/support/knowledgecenter/en/ssb27h_6.2.0/fa2ti_what_is_socket_connection.html (visited on 10/02/2019).
- [65] H. A. Yanco and J. Drury. Classifying human-robot interaction: An updated taxonomy. *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*. 2004. ISBN: 0-7803-8566-7. DOI: 10.1109/ICSMC.2004.1400763.

A SCRIPTS

Script for Right manipulator in YuMi (RAPID):

```

1  MODULE MainModule
2      !Picking box / faces targets
3      CONST robtarget basket:=[[368.04,413.24,138.77],[0.0157536,0.713784,
        0.700098,0.0112352],[1,3,1,4],[178.577,9E+09,9E+09,9E+09,9E+09,9E+09]];
4      CONST robtarget basketSlot:=[[182.67,203.11,143.84],[0.0158313,0.713717,
        0.700165,0.0112512],[1,2,0,4],[178.578,9E+09,9E+09,9E+09,9E+09,9E+09]];
5      CONST robtarget conveyor:=[[42.93,369.85,140.72],[0.0168502,0.713668,
        0.700191,0.0112373],[1,1,1,4],[178.722,9E+09,9E+09,9E+09,9E+09,9E+09]];
6      CONST robtarget faces:=[[182.60,441.86,127.45],[0.0167082,0.713684,
        0.700182,0.0110495],[1,3,1,4],[178.722,9E+09,9E+09,9E+09,9E+09,9E+09]];
7      !Classification targets
8      CONST robtarget pick:=[[172.60,
        267.07,48.83],[0.723561,0.00799609,0.690001,0.0171443],[1,0,2,4],[106.45,9E
        +09,9E+09,9E+09,9E+09,9E+09]];
9      CONST robtarget inspect:=[[197.33,241.88,195.06],[0.526606,0.479441,0.496266,
        0.49653],[0,0,1,4],[106.465,9E+09,9E+09,9E+09,9E+09,9E+09]];
10     CONST robtarget packing:=[[265.97,112.04,72.96],[0.683706,
        0.00338278,0.728887,0.0354852],[1,1,1,4],[165.945,9E+09,9E+09,9E+09,9E
        +09,9E+09]];
11     CONST robtarget buffer:=[[376.86,
        243.83,77.55],[0.786157,0.0170026,0.61683,0.0344767],[1,0,2,4],[103.333,9E
        +09,9E+09,9E+09,9E+09,9E+09]];
12     !Box faces targets
13     VAR robtarget F1:=[[251.15,45.99,111.40],[0.0361444,0.709161,
        0.703849,0.0195104],[1,2,0,4],[175.456,9E+09,9E+09,9E+09,9E+09,9E+09]];
14     VAR robtarget F5;
15     !Fitting targets
16     CONST robtarget F12_R :=[[454.09,54.20,154.83],[0.998033,0.0589704,
        0.0212458,0.00131195],[1,0,2,4],[136.196,9E+09,9E+09,9E+09,9E+09,9E
        +09]];
17     CONST robtarget F32_R :=[[447.16,
        79.05,174.37],[0.693291,0.0629267,0.0327608,0.717157],[1,0,0,4],[166.073,9E
        +09,9E+09,9E+09,9E+09,9E+09]];

```

```

18  CONST robtarget F43_R :=[[478.22,
      85.35,181.44],[0.674665,0.0806442,0.0524449,0.731829],[1,1,0,4],[166.077,9E
      +09,9E+09,9E+09,9E+09,9E+09]];
19  CONST robtarget F54_R :=[[295.89,7.95,194.29],[0.474717,0.50631,0.509383,
      0.508746],[1,1,0,4],[176.833,9E+09,9E+09,9E+09,9E+09,9E+09]];
20  CONST robtarget F65_R :=[[426.03,13.22,187.27],[0.262985,0.670715,
      0.62968,0.290659],[1,2,0,4],[174.995,9E+09,9E+09,9E+09,9E+09,9E+09]];
21  !Grasp from one side to avoid colision on final position
22  !Other Variables
23  VAR cameratarget mycameratarget;
24  VAR bool LoadPr:= TRUE;
25  CONST string myjob := "job1.job";
26  VAR robtarget p1;
27  VAR speeddata Vval := v200;
28  !Multitask synchronization variables
29  PERS tasks multi_task{2} := ["T_ROB_R"], ["T_ROB_L"];
30  VAR syncident sync0;
31  VAR syncident sync1;
32  VAR syncident sync2;
33  VAR syncident sync3;
34  VAR syncident sync4;
35  VAR syncident sync5;
36  VAR syncident sync6;
37  VAR syncident sync7;
38  VAR syncident sync8;
39  VAR syncident sync9;
40  VAR syncident sync10;
41  VAR syncident sync11;
42  VAR syncident sync12;
43  VAR syncident sync13;
44  VAR syncident sync14;
45  VAR syncident sync15;
46  VAR syncident sync16;
47  VAR syncident sync17;
48  VAR syncident sync18;
49  VAR syncident sync19;
50  VAR num gripperforce;
51  VAR string feedb;
52  VAR num parts;
53  VAR bool partsok;
54  !Sockets
55  VAR socketdev client_socket;

```

```

56  VAR string Tflag;
57  !Procedures Functions
58  PROC TRANSITION(num stage)
59      Idle:
60      SocketCreate client_socket;
61      SocketConnect client_socket, "192.168.125.208", 32200;
62      WHILE Tflag = "No" DO
63          SocketSend client_socket \Str:= NumToStr(stage,0);
64          SocketReceive client_socket \Str:= Tflag;
65          WaitTime(0.5);
66      ENDWHILE
67      SocketSend client_socket \Str:= NumToStr(100,0);
68      SocketClose client_socket;
69      IF Tflag="B" THEN
70          MoveJ Offs(basket, 0, 0, 100), Vval, fine, tool0;
71          MoveL basket, v50, fine, tool0;
72          g_GripIn \holdForce:=gripperforce;
73          MoveL Offs(basket, 0, 0, 100), Vval, fine, tool0;
74          MoveJ Offs(basketSlot, 0, 0, 100), Vval, fine, tool0;
75          MoveL basketSlot, v50, fine, tool0;
76          g_GripOut;
77          MoveJ Offs(basketSlot, 0, 0, 100), Vval, fine, tool0;
78          MoveL inspect, Vval, fine, tool0;
79          NOTIF(400);
80          Tflag:="No";
81          GOTO Idle;
82      ELSEIF Tflag="F" THEN
83          MoveJ Offs(faces, 0, 0, 100), Vval, fine, tool0;
84          MoveL faces, v50, fine, tool0;
85          g_GripIn \holdForce:=gripperforce;
86          MoveL Offs(faces, 0, 0, 100), Vval, fine, tool0;
87          MoveJ Offs(conveyor, 0, 0, 100), Vval, fine, tool0;
88          MoveL conveyor, v50, fine, tool0;
89          g_GripOut;
90          MoveJ Offs(conveyor, 0, 0, 100), Vval, fine, tool0;
91          MoveL inspect, Vval, fine, tool0;
92          NOTIF(500);
93          Tflag:="No";
94          GOTO Idle;
95      ENDIF
96      NOTIF(100);
97      Tflag := "No";

```

```

98     ENDPROC
99
100    PROC NOTIF (num stage)
101        SocketCreate client_socket;
102        SocketConnect client_socket, "192.168.125.208", 32200;
103        SocketSend client_socket \Str:= NumToStr(stage,0);
104        SocketReceive client_socket \Str:= feedb;
105        WaitTime(0.5);
106        SocketClose client_socket;
107    ENDPROC
108    PROC CLASSIFY()
109        MoveL inspect, Vval, fine, tool0;
110        CamReqImage RightCAM;
111        CamGetResult RightCAM, mycameratarget;
112        !vision inspect
113        !Model I found
114        If mycameratarget.val1=1 THEN
115            NOTIF(10);
116        !Model H found
117        ELSEIF mycameratarget.val2=1 THEN
118            NOTIF(11);
119        !Model F found
120        ELSEIF mycameratarget.val3=1 THEN
121            NOTIF(12);
122        ENDIF
123        !Align with picking point (elevated)
124        MoveL Offs(pick, 0, 0, 180), Vval, fine, tool0;
125        !Go to the paper
126        MoveL pick, v50, fine, tool0;
127        !Pick it
128        g_VacuumOn1;
129        WaitTime (2);
130        !Enable air blast
131        SetDO custom_DO_7,1;
132        !Lift slowly
133        MoveL Offs(pick, 0, 0, 20), v20, fine, tool0;
134        MoveL Offs(pick, 0, 0, 40), v50, fine, tool0;
135        MoveL Offs(pick, 0, 0, 180), v80, fine, tool0;
136        !Deactivate air blast
137        SetDO custom_DO_7,0;
138        !Read the command from backend and send it to pack or buffer
139        If feedb="pack" THEN

```

```

140         MoveJ Offs(packing, 0, 0, 50), v100, fine, tool0;
141         MoveL packing, v50, fine, tool0;
142     ELSEIF feedb="buffer" THEN
143         MoveJ Offs(buffer, 0, 0, 50), v100, fine, tool0;
144         MoveL buffer, v50, fine, tool0;
145     ENDIF
146     !Drop
147     g_VacuumOff1;
148     g_BlowOn1;
149     WaitTime (1);
150     g_BlowOff1;
151     !Lift arm wherever it is
152     p1 := CRobT();
153     MoveL Offs(p1, 0, 0, 100), v100, fine, tool0;
154 ENDPROC
155 PROC syncmove()
156     SyncMoveOn sync2, multi_task;
157     MoveL F12_R \ID:=10, v50, fine, tool0;
158     TRANSITION(1);
159     SyncMoveOff sync3;
160     UNDO
161         SyncMoveUndo;
162 ENDPROC
163 PROC syncmove2()
164     SyncMoveOn sync5, multi_task;
165     MoveL F32_R \ID:=20, v50, fine, tool0;
166     TRANSITION(2);
167     SyncMoveOff sync6;
168     UNDO
169         SyncMoveUndo;
170 ENDPROC
171 PROC syncmove3()
172     SyncMoveOn sync12, multi_task;
173     MoveL F43_R \ID:=30, v50, fine, tool0;
174     TRANSITION(3);
175     SyncMoveOff sync13;
176     UNDO
177         SyncMoveUndo;
178 ENDPROC
179 PROC syncmove4()
180     SyncMoveOn sync16, multi_task;
181     MoveL F54_R \ID:=40, v50, fine, tool0;

```

```

182     TRANSITION(4);
183     SyncMoveOff sync17;
184     UNDO
185         SyncMoveUndo;
186 ENDPROC
187 PROC Loadcam()
188     IF LoadPr=TRUE THEN
189         CamSetProgramMode RightCAM;
190         CamLoadJob RightCAM, "job1.job";
191         CamSetRunMode RightCAM;
192         LoadPr:=FALSE;
193     ENDIF
194 ENDPROC
195 PROC main()
196     !*****
197     !Box Assembly
198     !*****
199     F5 := Offs(F1,0,50,0);
200     gripperforce := 20;
201     Tflag := "No";
202     REPEAT:
203     TRANSITION(0);
204     NOTIF(300);
205     partsok := StrToVal(feedb, parts);
206     WaitSyncTask sync0, multi_task;
207     g_Init \maxSpd:=10 \holdForce:=10 \Calibrate;
208     g_SetForce 10;
209     g_SetMaxSpd 10;
210     MoveJ Offs(F1, 0, 0, 100), Vval, fine, tool0;
211     g_GripOut;
212     MoveL F1, v50, fine, tool0;
213     g_GripIn \holdForce:=gripperforce;
214     MoveL Offs(F1, 0, 0, 100), Vval, fine, tool0;
215     MoveJ RelTool(F12_R, 0, 50, 0), Vval, fine, tool0;
216     WaitSyncTask sync1, multi_task;
217     syncmove;
218     WaitSyncTask sync4, multi_task;
219     WaitSyncTask sync7, multi_task;
220     MoveL RelTool(F32_R, 70, 0, 50), Vval, fine, tool0;
221     WaitSyncTask sync8, multi_task;
222     MoveL RelTool(F32_R, 0, 0, 50), Vval, fine, tool0;
223     syncmove2;

```

```

224      WaitSyncTask sync9, multi_task;
225      MoveJ RelTool(F43_R, 50, 0, 0), v100, fine, tool0;
226      syncmove3;
227      g_GripOut;
228      MoveL RelTool(F43_R, 0, 0, 50), v100, fine, tool0;
229      WaitSyncTask sync10, multi_task;
230      MoveL RelTool(F43_R, 0, 300, 50), v100, fine, tool0;
231      MoveJ Offs(F5, 0, 0, 100), Vval, fine, tool0;
232      MoveL F5, v50, fine, tool0;
233      g_GripIn \holdForce:=gripperforce;
234      MoveL Offs(F5, 0, 0, 100), Vval, fine, tool0;
235      WaitSyncTask sync11, multi_task;
236      MoveJ RelTool(F54_R, 100, 0, 0), v100, fine, tool0;
237      syncmove4;
238      NOTIF(5);
239      g_GripOut;
240      MoveL RelTool(F54_R, 0, 0, 250), Vval, fine, tool0;
241      MoveJ inspect, Vval, fine, tool0;
242      WaitSyncTask sync14, multi_task;
243      WaitSyncTask sync15, multi_task;
244      !*****
245      !Classification
246      !*****
247      Loadcam;
248      FOR i FROM 1 TO parts DO
249          CLASSIFY;
250      ENDFOR
251      MoveL inspect, Vval, fine, tool0;
252      NOTIF(30);
253      WaitSyncTask sync18, multi_task;
254      WaitSyncTask sync19, multi_task;
255      NOTIF(200);
256      WaitTime(5);
257      GOTO REPEAT;
258      ERROR
259          IF ERRNO=ERR_SOCKET_TIMEOUT THEN
260              RETRY;
261          ELSEIF ERRNO=ERR_SOCKET_CLOSED THEN
262              RETRY;
263          ENDIF
264      ENDPROC

```

265 ENDMODULE

Script A.1. Script for Right manipulator in YuMi (RAPID)

Script for Left manipulator in YuMi (RAPID):

```

1  MODULE MainModule
2      !Faces targets
3      VAR robtarget F2:=[[217.50,270.42,97.01],[0.0378795,0.69461,
        0.717906,0.0263026],[1,2,1,4],[109.144,9E+09,9E+09,9E+09,9E+09,9E+09]];
4      VAR robtarget F3;
5      VAR robtarget F4;
6      VAR robtarget F6;
7      CONST robtarget HomePos:=[[36.36,252.47,230.83],[0.0230078,0.650791,
        0.758609,0.021297],[0,2,1,4],[103.371,9E+09,9E+09,9E+09,9E+09,9E+09]];
8      CONST robtarget packing_drop:=[[187.23,79.29,86.12],[0.101524,
        0.993924,0.0127726,0.0405671],[1,2,1,4],[149.436,9E+09,9E+09,9E+09,9E
        +09,9E+09]];
9      !Fitting targets
10     CONST robtarget F12_L:=[[364.28,28.48,373.22],[0.0448571,0.692267,
        0.720223,0.0057613],[1,1,0,4],[155.125,9E+09,9E+09,9E+09,9E+09,9E+09]];
11     CONST robtarget F32_L:=[[247.16,
        4.13,271.85],[0.709806,0.0450996,0.70216,0.033366],[1,1,1,5],[155.814,9E
        +09,9E+09,9E+09,9E+09,9E+09]];
12     CONST robtarget F43_L:=[[460.02,95.38,135.53],[0.694197,0.0088047,0.0374715,
        0.718755],[1,1,2,4],[172.968,9E+09,9E+09,9E+09,9E+09,9E+09]];
13     CONST robtarget F54_L
        :=[[279.09,84.09,241.05],[0.715549,0.0470472,0.696597,0.0229988],[1,1,
        1,4],[176.863,9E+09,9E+09,9E+09,9E+09,9E+09]];
14     CONST robtarget F65_L :=[[271.86,94.38,67.61],[0.343411,0.572924,0.627282,
        0.40043],[1,1,1,4],[135.686,9E+09,9E+09,9E+09,9E+09,9E+09]];
15     !Multitask synchronization variables
16     PERS tasks multi_task{2} := ["T_ROB_R"], ["T_ROB_L"];
17     VAR syncident sync0;
18     VAR syncident sync1;
19     VAR syncident sync2;
20     VAR syncident sync3;
21     VAR syncident sync4;
22     VAR syncident sync5;
23     VAR syncident sync6;
24     VAR syncident sync7;
25     VAR syncident sync8;
26     VAR syncident sync9;
27     VAR syncident sync10;

```



```

28  VAR syncident sync11;
29  VAR syncident sync12;
30  VAR syncident sync13;
31  VAR syncident sync14;
32  VAR syncident sync15;
33  VAR syncident sync16;
34  VAR syncident sync17;
35  VAR syncident sync18;
36  VAR syncident sync19;
37  VAR num gripperforce;
38  VAR speeddata Vval := v200;
39  !Procedures Functions
40  PROC syncmove()
41      SyncMoveOn sync2, multi_task;
42      MoveL F12_L \ID:=10, v50, fine, tool0;
43      SyncMoveOff sync3;
44      UNDO
45          SyncMoveUndo;
46  ENDPROC
47  PROC syncmove2()
48      SyncMoveOn sync5, multi_task;
49      MoveL F32_L \ID:=20, v50, fine, tool0;
50      SyncMoveOff sync6;
51      UNDO
52          SyncMoveUndo;
53  ENDPROC
54  PROC syncmove3()
55      SyncMoveOn sync12, multi_task;
56      MoveL F43_L \ID:=30, v50, fine, tool0;
57      SyncMoveOff sync13;
58      UNDO
59          SyncMoveUndo;
60  ENDPROC
61  PROC syncmove4()
62      SyncMoveOn sync16, multi_task;
63      MoveL F54_L \ID:=40, v50, fine, tool0;
64      SyncMoveOff sync17;
65      UNDO
66          SyncMoveUndo;
67  ENDPROC
68  PROC main()
69      !*****

```

```

70      !Box Assembly
71      !*****
72      F6 := Offs(F2,0,50,80);
73      F4 := Offs(F2,0,50,0);
74      F3 := Offs(F2,0,100,80);
75      gripperforce := 20;
76      REPEAT:
77      WaitSyncTask sync0, multi_task;
78      g_Init \maxSpd:=10 \holdForce:=10 \Calibrate;
79      g_SetForce 10;
80      g_SetMaxSpd 10;
81      MoveJ HomePos, Vval, fine, tool0;
82      MoveL Offs(F2, 0, 0, 100), Vval, fine, tool0;
83      g_GripOut;
84      MoveL F2,v50,fine,tool0;
85      g_GripIn \holdForce:=grripperforce;
86      MoveL Offs(F2, 0, 0, 100), Vval, fine, tool0;
87      WaitSyncTask sync1, multi_task;
88      MoveJ RelTool(F12_L, 0, 60, 0), Vval, fine, tool0;
89      syncmove;
90      WaitSyncTask sync4, multi_task;
91      g_GripOut;
92      MoveL RelTool(F12_L, 0, 0, 40), v100, fine, tool0;
93      WaitSyncTask sync7, multi_task;
94      MoveJ RelTool(F12_L, 150, 0, 0), Vval, fine, tool0;
95      MoveL F3,Vval,fine,tool0;
96      g_GripIn \holdForce:=grripperforce;
97      MoveL Offs(F3, 0, 0, 50), Vval, fine, tool0;
98      MoveJ RelTool(F32_L, 50, 0, 0), Vval, fine, tool0;
99      WaitSyncTask sync8, multi_task;
100     syncmove2;
101     g_GripOut;
102     MoveL RelTool(F32_L, 0, 250, 0), Vval, fine, tool0;
103     MoveJ Offs(F4, 0, 0, 100), Vval, fine, tool0;
104     MoveL F4,v50,fine,tool0;
105     g_GripIn \holdForce:=grripperforce;
106     WaitSyncTask sync9, multi_task;
107     MoveL Offs(F4, 0, 0, 200), Vval, fine, tool0;
108     MoveJ RelTool(F43_L, 200, 0, 0), Vval, fine, tool0;
109     syncmove3;
110     !WaitSyncTask sync11, multi_task;
111     WaitSyncTask sync10, multi_task;

```

```

112      MoveL RelTool(F43_L, 150, 0, 0), v100, fine, tool0;
113      MoveL RelTool(F43_L, 150, 0, 0 \Rx:= 50), v20, fine, tool0;
114      MoveL RelTool(F43_L, 150, 0, 0 \Rx:=50 \Rz:= 120), v20, fine, tool0;
115      MoveL RelTool(F43_L, 150, 0, 0 \Rx:=10 \Rz:= 120), v20, fine, tool0;
116      WaitSyncTask sync11, multi_task;
117      MoveJ RelTool(F54_L, 0, 50, 0), Vval, fine, tool0;
118      syncmove4;
119      WaitSyncTask sync14, multi_task;
120      MoveJ RelTool(packing_drop, 0, 0, 140), v100, fine, tool0;
121      MoveL packing_drop, v50, fine, tool0;
122      g_GripOut;
123      MoveL RelTool(packing_drop, 0, 0, 100), Vval, fine, tool0;
124      WaitSyncTask sync15, multi_task;
125      MoveJ HomePos, v100, fine, tool0;
126      WaitSyncTask sync18, multi_task;
127      !*****
128      !Close box
129      !*****
130      MoveJ Offs(F6, 0, 0, 70), Vval, fine, tool0;
131      MoveL F6, v50, fine, tool0;
132      g_GripIn \holdForce:=gripperforce;
133      MoveL Offs(F6, 0, 0, 70), Vval, fine, tool0;
134      MoveJ Offs(F65_L, 0, 0, 50), Vval, fine, tool0;
135      MoveL F65_L, v50, fine, tool0;
136      g_GripOut;
137      MoveL RelTool(F65_L, 0, 0, 50), v50, fine, tool0;
138      MoveL HomePos, Vval, fine, tool0;
139      WaitSyncTask sync19, multi_task;
140      GOTO REPEAT;
141      ENDPROC
142  ENDMODULE

```

Script A.2. Script for Left manipulator in YuMi (RAPID)

Application backend (Python 3):

```

1  # *****
2  #
3  # Tampere University
4  # Future Automation Systems and Technologies Laboratory
5  # FAST-LAB
6  #
7  # *****
8  # Design and implementation of a human-robot collaborative

```

```

 9 # assembly workstation in a modular robotized production line
10 #
11 # This script acts as orchestrator for the YuMi collaborative
12 # work station. It runs 2 threads.
13 # Thread 1: HTTP Server - Using FLASK-SOCKET IO
14 # Thread 2: Socket Server - Using Socket
15 #
16 # The decorators for FLASK-SOCKET IO serve a web based HMI
17 # with wireless connection to this server
18 #
19 # *****
20 # V0:06.10.2019 - RB
21 # *****
22 # Import libraries - Please check dependencies
23 from flask_socketio import SocketIO, emit
24 from flask import Flask, render_template, url_for, copy_current_request_context,
    request
25 from random import random
26 from time import sleep
27 from threading import Thread, Event
28 from flask_cors import CORS
29 from sqlalchemy import Column, Integer, String, Text, DateTime, Boolean
30 from flask_sqlalchemy import SQLAlchemy
31 from requests.adapters import HTTPAdapter
32 from requests.exceptions import ConnectionError
33 import socket
34 import requests
35 import json
36 import datetime
37
38 # *****
39 # Script setup
40 # *****
41 # Method to optimize event report for event subscribers
42 adapterx=HTTPAdapter(max_retries=2)
43 sess=requests.Session()
44 sess.mount('http://', adapterx)
45
46 HOST = '0.0.0.0' # 0.0.0.0, use ethernet device IP for socket server - Setup
    in YuMi as 192.168.125.208
47 PORTS = 32200 # Port to serve Socket requests from YuMi
48 PORTF = 32100 #Port to serve Flask - HTTP requests

```

```

49
50 #Create FLASK instance
51 app = Flask(__name__)
52 #CORS for cross origin - Server hosting the API in a different location
    while debug (not the HMI tablet)
53 CORS(app)
54 # Database file for event subscription and new instance
55 app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///tmp/subs.db'
56 db = SQLAlchemy(app)
57 # parameters for subscriber table
58
59 class Subscriber(db.Model):
60     __tablename__ = 'subscriber'
61
62     id = Column(Integer(), primary_key=True, default=None)
63     created = Column(DateTime())
64     destUrl = Column(Text())
65     eventID = Column(Text())
66
67     @property
68     def subscription_info_json(self):
69         return json.loads(self.destUrl)
70
71     @subscription_info_json.setter
72     def subscription_info_json(self, value):
73         self.destUrl = json.dumps(value)
74
75 # Create subscriber table on /tmp.subs.db
76
77
78 db.create_all()
79
80 # Settings for future production server mode - use waitress
81 app.config['DEBUG'] = True
82
83 # Turn the flask instance into a flask-socketio
84 socketio = SocketIO(app)
85
86 # Check if this is actually needed
87 thread = Thread()
88 thread_stop_event = Event()
89

```

```

90 # *****
91 # Variables for multi threading
92 # *****
93 #evTr auxiliar variable holding the message to the cobot while is connected
94 evTr = "No"
95 # Phone contains the number of phones per style to be packed in the order
96 phones = [0, 0, 0]
97 # Variable describing order id
98 OrID = "0"
99 # Variable describing the amount of products in the basket
100 Basqty = 0
101 # Flag variable to report if YuMi is on Idle state
102 IdleState = True
103
104 # Socket server. Includes and instance of the class Thread from threading
    library
105 class RobotThread(Thread):
106     #Global variables exchanged between servers
107     global evTr
108     global phones
109     global IdleState
110     global Basqty
111     # Function to handle sockets server requests from YuMi
112     def socketHandler(self):
113         # Global variables exchanged between servers
114         global evTr
115         global phones
116         global IdleState
117         global Basqty
118         # Function to update the product count while classifying
119         # and return a signal to YuMi to pack or buffer
120         def evalproduct (product):
121             prod = int(product)
122             if prod > 0:
123                 prod = 1
124                 mess = "pack"
125             else:
126                 mess = "buffer"
127             return mess, prod
128         # If there is not order to stop the thread, execute it
129         while not thread_stop_event.isSet():
130             # Exception handling for socket instance

```

```

131 with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
132     #Start socket server
133     s.bind((HOST, PORTS))
134     s.listen()
135     print('WebSockets_server_enabled')
136     # Capture every requests from addr on conn
137     conn, addr = s.accept()
138     # Exception handling for socket connection request
139     with conn:
140         print('Connected_by', addr)
141         # evTr keeps alive the socket connection while bouncing
142         "No"
143         while evTr == "No":
144             # read data from connection request
145             socketdata = conn.recv(1024)
146             # If data is not empty
147             if socketdata:
148                 # Convert data to integer
149                 socketdata = int(socketdata.decode('utf8'))
150                 # Evaluate action code: 0-6,10-12,30,100,200,300
151                 if socketdata == 0:
152                     IdleState = True
153                     process = "Idle"
154                     stage = "Waiting_for_start_command"
155                 elif 0 < socketdata < 5:
156                     IdleState = False
157                     process = "Case_assembly"
158                     switcher = {
159                         1: 'Fastening_pieces_1_2',
160                         2: 'Fastening_pieces_1_to_3',
161                         3: 'Fastening_pieces_1_to_4',
162                         4: 'Fastening_pieces_1_to_5',
163                     }
164                     stage = switcher.get(socketdata, "Unknown_step")
165                 elif socketdata == 5:
166                     process = "Classifying_and_packing"
167                     stage = "Visual_inspection"
168                     evTr = "Tr"
169                 elif socketdata == 6:
170                     process = "Collecting_external_materials"
171                     stage = "Picking"
172                     evTr = "Tr"

```

```

172         # If more models are created, please insert the
           action codes from here
173     elif socketdata == 10:
174         process = "Classifying_and_packing"
175         stage = "Phone_model_I"
176         evTr, phones[0] = evalproduct(phones[0])
177     elif socketdata == 11:
178         process = "Classifying_and_packing"
179         stage = "Phone_model_H"
180         evTr, phones[1] = evalproduct(phones[1])
181     elif socketdata == 12:
182         process = "Classifying_and_packing"
183         stage = "Phone_model_F"
184         evTr, phones[2] = evalproduct(phones[2])
185     # to here
186     elif socketdata == 30:
187         process = "Classifying_and_packing"
188         stage = "Sealing_package"
189         evTr = "Tr"
190     elif socketdata == 100:
191         process = "Case_assembly"
192         stage = "Picking_pieces"
193         evTr = "Tr"
194     elif socketdata == 200:
195         process = "Process_completed"
196         stage = "Products_ready_to_dispatch"
197         socketio.emit('finish', namespace='/HMI')
198         socketio.emit('video', {'videon': 0}, namespace='/HMI')
199         socketio.emit('orderid', {'orderid': "Not_received"},
           namespace='/HMI')
200         evTr = "Tr"
201         PNotif("collabdone", OrID)
202         IdleState = True
203     elif socketdata == 400:
204         #Basket picked send event update
205         PNotif('basketloaded', OrID)
206         process = "Idle"
207         stage = "Waiting_for_start_command"
208         evTr = "Tr"
209     elif socketdata == 500:
210         # Faces picked
211         PNotif('facesloaded', "")

```



```

212         process = "Idle"
213         stage = "Waiting_for_start_command"
214         evTr = "Tr"
215         elif socketdata == 300:
216             # Reports the amount of products in the basket
217             evTr = str(Basqty)
218             print('Recieved:', socketdata)
219             socketio.emit('status', {'stage': stage, 'process': process},
220                           namespace='/HMI')
221             conn.sendall(evTr.encode('utf8'))
222             print("Sent:", evTr)
223             # For picking material at AMIR
224             if evTr == "B":
225                 socketio.emit('status', {'stage': 'Picking_product_basket', 'process': '
                Receive_supplies'},
226                               namespace='/HMI')
227             if evTr == "F":
228                 socketio.emit('status', {'stage': 'Picking_case_faces', 'process': 'Receive
                _supplies'},
229                               namespace='/HMI')
230             evTr = "No"
231             conn.close()
232             if 0 < socketdata < 5:
233                 socketio.emit('video', {'videon': socketdata}, namespace='/HMI')
234                 print("video_requested:", socketdata)
235                 socketio.emit('robot', namespace='/HMI')
236                 sleep(0.2)
237         def run(self):
238             self.socketHandler()
239
240         # *****
241         # Functions and headers
242         # *****
243         # Header for AMIR requests
244
245
246         def headers():
247             Auth = "Basic_
                ZGIzdHJpYnV0b3I6NjJmMmYwZjFIZmYxMGQzMtUyYzk1ZjZmMDU5NjU3NmU0ODJiYjhl
                =="
248             header={

```

```

249         "Authorization": Auth,
250         "Accept": "application/json",
251         "AcceptEncoding": "gzip,_deflate",
252         "AcceptLanguage": "enUS",
253         "ContentType": "application/json",
254     }
255     return header
256 # Headers for notifications
257
258
259 def notiheaders():
260     header={
261         "ContentType": "application/json",
262     }
263     return header
264
265 # Function to send notifications
266
267
268 def PNotif (evid, InfoEv):
269     items = Subscriber.query.filter(Subscriber.eventID == evid).all()
270     for _item in items:
271         try:
272             notification = sess.post(_item.destUrl, headers=notiheaders(),
273                                     json={"class": "eventNotification", "eventID": evid, "
                                         Info": InfoEv}, timeout=0.1)
274             print("Client:_", _item.destUrl, "Response:_", notification.status_code)
275         except ConnectionError as er:
276             print('Error_sending_notification', er)
277 # Function to subscribe to an event. Stores url and event in tmp/subs.db
278
279 def subscribeev(eventname, sub_info):
280
281     # Does not check if info is correct. If the client is not on db return
282     # code 418 Im a teapot, otherwise 200
283     item = Subscriber.query.filter(Subscriber.destUrl == sub_info, Subscriber.eventID
284                                   == eventname).first()
285     if not item:
286         item = Subscriber()
287         item.created = datetime.datetime.utcnow()
288         item.destUrl = sub_info
289         item.eventID = eventname

```

```

288         db.session.add(item)
289         db.session.commit()
290         print("New_client_subscribed_to", eventname, ":", sub_info)
291         bodres = {'Client': sub_info, 'Event': eventname, 'Status': 'Subscribed'}
292         return bodres, 200
293     else:
294         bodres = {'Client': sub_info, 'Event': eventname, 'Status': 'Failed'}
295         return bodres, 418
296
297 # Function to unsubscribe to an event. Delete the listing from tmp/subs.db
298
299
300 def unsubscribeev(eventname, sub_info):
301
302     # Does not check if info is correct. If the client is not on db return
303     # code 418 Im a teapot, otherwise 200
304     item = Subscriber.query.filter(Subscriber.destUrl == sub_info, Subscriber.eventID
305     == eventname).first()
306     if item:
307         db.session.delete(item)
308         db.session.commit()
309         print("Client_unsubscribed_from", eventname, ":", sub_info)
310         bodres = {'Client': sub_info, 'Event': eventname, 'Status': 'Unsubscribed'}
311         return bodres, 200
312     else:
313         bodres = {'Client': sub_info, 'Event': eventname, 'Status': 'Failed'}
314         return bodres, 418
315
316 # *****
317 # Web HMI Server
318 # *****
319
320 # Endpoint to return the HTML frontend
321
322 @app.route('/')
323 def index():
324     # only by sending this page first will the client be connected to the
325     # socketio instance
326     return render_template('HomeUI.html')
327
328 # *****

```

```

327 # APIs - Events
328 # *****
329 # Endpoint to subscribe for Basket Loaded notifications
330
331
332 @app.route('/events/basketloaded/notifs', methods=['POST', 'DELETE'])
333 def basketloadedsubs():
334     sub_info = request.get_json()
335     sub_info = sub_info["destUrl"]
336     if request.method == 'POST':
337         resb, resbc = subscribeev("basketloaded", sub_info)
338     if request.method == 'DELETE':
339         resb, resbc = unsubscribeev("basketloaded", sub_info)
340     return json.dumps(resb), resbc, {'ContentType': 'application/json'}
341
342
343 # Endpoint to subscribe for Faces Loaded notifications
344
345
346 @app.route('/events/facesloaded/notifs', methods=['POST', 'DELETE'])
347 def facesloadedsubs():
348     sub_info = request.get_json()
349     sub_info = sub_info["destUrl"]
350     if request.method == 'POST':
351         resf, resfc = subscribeev("facesloaded", sub_info)
352     if request.method == 'DELETE':
353         resf, resfc = unsubscribeev("facesloaded", sub_info)
354     return json.dumps(resf), resfc, {'ContentType': 'application/json'}
355
356 # Endpoint to subscribe for Collaborative Process Done notifications
357
358
359 @app.route('/events/collabdone/notifs', methods=['POST', 'DELETE'])
360 def collabdonesubs():
361     sub_info = request.get_json()
362     sub_info = sub_info["destUrl"]
363     if request.method == 'POST':
364         resc, rescc = subscribeev("collabdone", sub_info)
365     if request.method == 'DELETE':
366         resc, rescc = unsubscribeev("collabdone", sub_info)
367     return json.dumps(resc), rescc, {'ContentType': 'application/json'}
368

```

```

369
370 # *****
371 # APIs - Services
372 # *****
373 # Endpoint to pick basket
374
375 @app.route('/services/pickbasket', methods=['POST'])
376 def pickb():
377     global evTr
378     global IdleState
379     global OrID
380     global Basqty
381     if IdleState:
382         bask_info = request.get_json()
383         OrID = bask_info["orderID"]
384         Basqty = bask_info["units"]
385         evTr = "B"
386         socketio.emit('orderid', {'orderid': OrID}, namespace='/HMI')
387         return json.dumps({'success': True}), 200, {'ContentType': 'application/json'}
388     else:
389         return json.dumps({'success': False}), 403, {'ContentType': 'application/json'}
390     # only by sending this page first will the client be connected to the
391     # socketio instance
392
393 # Endpoint to pick faces
394
395 @app.route('/services/pickfaces', methods=['POST'])
396 def pickf():
397     global evTr
398     global IdleState
399     if IdleState:
400         evTr = "F"
401         return json.dumps({'success': True}), 200, {'ContentType': 'application/json'}
402     else:
403         return json.dumps({'success': False}), 403, {'ContentType': 'application/json'}
404
405 # Endpoint to enable collaborative robot
406
407
408 @app.route('/services/enablecollab', methods=['POST'])
409 def enablecollab():

```

```

410     global phones
411     collab_info = request.get_json()
412     phones = collab_info["order"]
413     if len(phones)!=3 or not IdleState:
414         return json.dumps({'success': False}), 403, {'ContentType': 'application/json'}
415     else:
416         socketio.emit('idle', namespace='/HMI')
417         return json.dumps({'success': True}), 200, {'ContentType': 'application/json'}
418
419
420 # *****
421 # HMI Interaction
422 # *****
423 # Event to start the process. Orders transition through evTr
424
425
426 @socketio.on('next', namespace='/HMI')
427 def handle_my_custom_event(self):
428     global evTr
429     evTr = "Tr"
430
431 # Default event when a web browser sends GET to the Flask server
432
433
434 @socketio.on('connect', namespace='/HMI')
435 def test_connect():
436     global thread
437     print('Client_connected')
438     # Start the secondary thread - Sockets server
439     if not thread.isAlive():
440         print("Starting_Thread")
441         robotthread = RobotThread()
442         robotthread.start()
443
444 # Provisional event to detect disconnection of HMI client
445
446 @socketio.on('disconnect', namespace='/HMI')
447 def test_disconnect():
448     print('Client_disconnected')
449
450 # Event to order AMIR to bring products
451

```

```

452
453 @socketio.on('supply', namespace='/HMI')
454 def supplyproduct(self):
455     supplyres = requests.post("http://192.168.100.103/api/v2.0.0/mission_queue",
456                               headers=headers(), json={"mission_id": "ad1dd7086b5911e9a3aa94
457                               c6911e7b24", "message": "start", "priority": 1})
458     print("Response: ", supplyres.status_code)
459
460 # Event to order AMIR to leave the collaborative workstation
461
462 @socketio.on('pickup', namespace='/HMI')
463 def pickupproduct(self):
464     pickupres = requests.post("http://192.168.100.103/api/v2.0.0/mission_queue",
465                               headers=headers(), json={"mission_id": "02f4a18a6b5b11e9a3aa94c6911e7b24
466                               ", "message": "start", "priority": 1})
467     print("Response: ", pickupres.status_code)
468
469 # *****
470 # Starting point
471 # *****
472
473 if __name__ == '__main__':
474     # Start HTTP server in port 32100
475     socketio.run(app, host=HOST, port=PORTF)

```

Script A.3. Application backend (Python 3)

Human Machine Interface (HTML):

```

1 <!DOCTYPE html>
2 <html>
3 <title>TUNI Cobots FASTLab</title>
4 <meta charset="UTF8">
5 <meta name="viewport" content="width=devicewidth, _initialscale=1">
6 <link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
7 <link rel="stylesheet" href="//maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.
8   min.css">
9 <script src="//code.jquery.com/jquery3.3.1.min.js"></script>
10 <script type="text/javascript" src="//cdnjs.cloudflare.com/ajax/libs/socket.io/1.7.3/
11   socket.io.min.js"></script>
12 <script type="text/javascript">
13     $(document).ready(function(){

```

```

12     var socket = io.connect('http://' + document.domain + ':' + location.port + '/HMI');
13     var vid = document.getElementById("videodemo");
14     var numbers_received = [];
15     $('#DButton').click(function(){
16         socket.emit('next', namespace='/HMI');
17         document.getElementById("DButton").disabled=true;
18     })
19     $('#SButton').click(function(){
20         $('#task').html('Picking pieces');
21         $('#process').html('Case assembly');
22         socket.emit('next', namespace='/HMI');
23         document.getElementById("DButton").hidden=false;
24     })
25     $('#Supply').click(function(){
26         socket.emit('supply', namespace='/HMI');
27         document.getElementById("Pickup").disabled=false;
28         document.getElementById("Supply").disabled=true;
29     })
30     $('#Pickup').click(function(){
31         socket.emit('pickup', namespace='/HMI');
32         document.getElementById("Pickup").disabled=true;
33         document.getElementById("Supply").disabled=false;
34     })
35     //Socket details from server
36     socket.on('status', function (msg) {
37         $('#task').html(msg.stage);
38         $('#process').html(msg.process);
39         document.getElementById("DButton").disabled=false;
40     })
41     socket.on('orderid', function (msg) {
42         $('#orderid').html("Order_ID:_" + msg.orderid);
43     })
44     socket.on('idle', function (msg) {
45         document.getElementById("SButton").disabled=false;
46     })
47     socket.on('robot', function () {
48         document.getElementById("DButton").disabled=true;
49         document.getElementById("SButton").disabled=true;
50     })
51     socket.on('finish', function () {
52         document.getElementById("DButton").hidden=true;
53         document.getElementById("SButton").disabled=false;

```



```

54
55     })
56     socket.on('video', function (msg) {
57         vid.src = "../static/Seq"+msg.videon+".mp4";
58         vid.load();
59     })
60 });
61 </script>
62 <body class="w3content" style="maxwidth:1300px">
63
64 <! First Grid: Logo & About >
65 <div class="w3row">
66     <div class="w3quarter_w3white_w3container_w3center" style="height:800px">
67         <div class="w3padding64">
68             
69             <h2>FASTLAB</h2>
70             <p>Faculty of Engineering and Natural Sciences</p>
71         </div>
72         <div class="w3padding64">
73             <a href="#" class="w3button_w3white_w3block_w3hovergrey_w3padding16">
74                 Home</a>
75             <a href="#about" class="w3button_w3white_w3block_w3hoverindigo_w3padding
76                 16">MIR control Schedule</a>
77             <a href="https://www.tuni.fi/en/research/fastlab" class="w3button_w3white_w3
78                 block_w3hoverbrown_w3padding16">FASTLAB</a>
79         </div>
80     </div>
81 <div class="w3quarter_w3indigo_w3container" style="height:800px">
82     <div class="w3padding64_w3center">
83         <h2>Collaborative Assembly Process</h2>
84         
86         <div class="w3centralalign_w3paddinglarge">
87             <p>Current task:</p>
88             <h3 class="w3padding8_w3center" id="task">Waiting</h3>
89             <p>Process:</p>
90             <h3 class="w3padding8_w3center" id="process">Waiting for robot</h3>
91             <input type="button" class="w3button_w3green_w3center_w3padding8" id="
92                 SButton" value="Start" disabled="true"></input>
93         </div>
94     </div>
95 </div>

```

```

90     </div>
91     <div class="w3half_w3lightgrey_w3container" style="height:800px">
92         <div class="w3padding32_w3center">
93             <h1>Process Execution</h1>
94             <div class="w3leftalign_w3paddinglarge">
95                 <video id= "videodemo" width="570" height="480" autoplay loop muted>
96                     <source src="./static/Seq0.mp4" type="video/mp4">
97                 </video>
98                 <button type="submit" class="w3button_w3green_w3right" id="DButton"
99                     hidden="false" >Done!</button>
100             <h3 class="w3padding8_w3center" id="orderid">Order ID: Not received</h3>
101         </div>
102     </div>
103 </div>
104
105 <!-- Second Grid: Work & Resume -->
106 <div class="w3row">
107     <div class="w3half_w3lightgrey_w3container" style="height:800px">
108         <div class="w3padding32_w3center">
109             <h1>Mobile robot control</h1>
110             <div class="w3leftalign_w3paddinglarge">
111                 <button type="submit" class="w3button_w3blue_w3left" id="Supply" disabled
112                     ="true" >Call AMIR</button>
113                 <button type="submit" class="w3button_w3blue_w3left" id="Pickup" >Release
114                     AMIR</button>
115             </div>
116         </div>
117     </div>
118 </div>
119 <div class="w3half_w3indigo_w3container" style="minheight:500px" id="about">
120     <div class="w3padding32_w3center">
121         <h2>Collaborative Workstation</h2>
122         <p>Work Schedule</p>
123         <div class="w3container_w3responsive">
124             <table class="w3table">
125                 <tr>
126                     <th>Date</th>
127                     <th>Operator</th>
128                     <th>Shift time</th>

```

```

129         <td>Carlos</td>
130         <td>6H</td>
131     </tr>
132     <tr>
133         <td>05/07/2019</td>
134         <td>Antti</td>
135         <td>8H</td>
136     </tr>
137     <tr class="w3white">
138         <td>06/07/2019</td>
139         <td>Mario</td>
140         <td>7H</td>
141     </tr>
142 </table>
143 </div>
144 </div>
145 </div>
146 </div>
147
148 <!-- Third Grid: Swing By & Contact -->
149 <div class="w3row" id="contact">
150     <div class="w3darkgrey_w3container_w3center" style="height:200px">
151         <div class="w3padding5">
152             <h1>Swing By</h1>
153         </div>
154         <div class="w3padding16">
155             <p>Tamepere University</p>
156             <p>Hervanta Campus Korkeakoulunkatu 5 Ri208</p>
157             <p>33720 Tampere, Finland</p>
158         </div>
159     </div>
160 </div>
161
162 <!-- Footer -->
163 <footer class="w3container_w3black_w3padding16">
164     <p>Powered by <a href="https://www.w3schools.com/w3css/default.asp" target="
        _blank">w3.css</a></p>
165 </footer>
166
167 </body>
168 </html>

```

Script A.4. Human Machine Interface - Frontend (HTML)

Plans conveyor logic (ST):

```

1  VAR_GLOBAL
2      Response:STRING(8):='Unloaded';
3      CNVID:STRING(8):='plans';
4  END_VAR
5
6  PROGRAM LOAD_PLANS
7  VAR
8      (** Locals here **)
9  END_VAR
10     IF S1 = true AND S2 = false THEN
11         DIRECTION := TRUE;
12         STOP := TRUE;
13         START := TRUE;
14         WHILE S2 = false DO
15             wait (50);
16         END_WHILE
17         START := FALSE;
18         STOP := FALSE;
19         DIRECTION := FALSE;
20         Response := 'Loaded';
21         REST_PUBLISH(CNVStatus);
22     END_IF
23 END_PROGRAM
24
25 PROGRAM UNLOAD_PLANS
26 VAR
27
28     (** Locals here **)
29 END_VAR
30     IF S2 = false AND Response = 'Loaded' THEN
31         Response := 'Unloaded';
32         REST_PUBLISH(CNVStatus);
33     END_IF
34
35 END_PROGRAM

```

Script A.5. Plans conveyor logic (ST)**Pallet conveyor logic (ST):**

```

1  VAR_GLOBAL
2      Response:STRING(8):='Unloaded';
3      CNVID:STRING(8):='pallet';

```

```

4  END_VAR
5
6  PROGRAM LOAD_PALLET
7  VAR
8
9      (** Locals here **)
10 END_VAR
11     IF S3 = true AND Response = 'Unloaded' THEN
12     DIRECTION := FALSE;
13     STOP := TRUE;
14     START := TRUE;
15     WHILE S1 = false DO
16         wait (50);
17     END_WHILE
18     STOP := FALSE;
19     START := FALSE;
20     DIRECTION := FALSE;
21     Response := 'Loaded';
22     REST_PUBLISH(CNVStatus);
23 END_IF
24
25 END_PROGRAM
26
27 PROGRAM UNLOAD_PALLET
28 VAR
29     (** Locals here **)
30 END_VAR
31     IF S1 = true AND S2 = false THEN
32     DIRECTION := TRUE;
33     STOP := TRUE;
34     START := TRUE;
35     WHILE S2 = false DO
36         wait (50);
37     END_WHILE
38     START := FALSE;
39     STOP := FALSE;
40     DIRECTION := FALSE;
41     WHILE S2 = true DO
42         wait (50);
43     END_WHILE
44     Response := 'Unloaded';
45     REST_PUBLISH(CNVStatus);

```

```
46     END_IF  
47 END_PROGRAM
```

Script A.6. Pallet conveyor logic (ST)